ICT IP Project

Deliverable D5.3.1

CHOReOS IDRE 1st Version User Manual

http://www.choreos.eu

LaTeX template v. 1.16

Project Number	:	FP7-257178
Project Title	:	CHOReOS Large Scale Choreographies for the Future Internet

Deliverable Number	:	D5.3.1
Title of Deliverable	:	CHOReOS IDRE 1st Version User Manual
Nature of Deliverable	:	Report + Prototype
Dissemination level	:	Public
Licence	:	Creative Commons Attribution 3.0 License
Version	:	1.14
Contractual Delivery Date	:	October 2012
Actual Delivery Date	:	November 2012
Contributing WP	:	WP5
Editor(s)	:	Amira Ben Hamida (EBM)
Author(s)	:	Amira Ben Hamida (EBM), Julien Lesbegueries (EBM), Fabio Kon (USP), Carlos Eduardo dos Santos (USP), Leonardo Ferreira Leite (USP), Nelson Lago (USP), Daniel Batista (USP), Ivanilton Polato (USP), Apostolos Zarras (UOI), Dionysis Athanasopoulos (UOI), Panos Vassiliadis (UOI), Spyros Lalis (VTRIP), Konstantinos Lekkas (VTRIP), Thanassis Parathyras (VTRIP), Nikolaos Georgantas (Inria), Valérie Issarny (Inria), Animesh Pathak (Inria), Sara Hachem (Inria), Sandrine Bauche (Inria), Pierre Chatel (Thalès), Rokas Bartkevičius (NME), Antonello Calabrò (CNR), Guglielmo De Angelis (CNR), Alberto Ribolini (CNR), Midhat Ali (UNICAM), Francesco De Angelis (UNICAM)
Reviewer(s)	:	Daniele Dell'Aglio (Cefriel)

Abstract

D5.3.1 is the first prototype deliverable of the CHOReOS Work-Package 5. It describes the CHOReOS Integrated Development and Runtime Environment (IDRE) and the user manual for its components. The CHOReOS IDRE is available in the OW2 CHOReOS repository and is open source. The current document briefly describes the delivered platform; it contains the cartography of the IDRE, an integrated vision of the different components taking part in the CHOReOS scenario, as well as the user manuals for each component. D5.3.1 is delivered at the same time as the Test Bed Implementation D5.4 and the second version of the integration plan D5.7.2.

Keyword List

Implementation, User manual, Integrated Development and Runtime Environment, Prototype, User Manual, extensible Service Discovery, executable Service Bus, extensible Service Access, EasyESB, Generic Application, Abstraction Oriented Service Base Management, Cloud and Grid.

Document History

Version	Changes	Author(s)
1.0	Initial commits of the deliverable on SVN	Linagora
1.1	Inputs for the components user manual	USP, Inria, UOI, CNR, Unicam, Linagora, VTRIP
1.2	Inputs for introduction, abstract, conclusion	Linagora

Document Reviews

Review	Date	Ver.	Reviewers	Comments
Outline	August. 3rd	1.0	Outline	na
Draft	October 21st	2.0	Valérie Issarny	Comments sent by mail.
QA	October 30 st	3.0	Daniele Dell'Aglio	Comments sent by mail and document annotation.
PTC	November 15th	4.0	na	na

Glossary, acronyms & abbreviations

Item	Description
API	Application Programming Interface
AOSMB	Abstraction Oriented Service Base Management
BC	Binding Component
BSM	Business Service Monitoring
CSDL	Client/Server Description Language
DL	Deliverable Leader
DOW	Description of Work
DPWSDL	DPWS Description Language
DSB	Distributed Service Bus
GA	Generic Application
GUI	Graphical User Interface
IAC	Industrial Advisory Committee
IDRE	Integrated Development and Runtime Environment
IOTS	Internet of Things Services
JBI	Java Business Integration
JMSDL	JMS Description Language
JSDL	JavaSpaces Description Language
Q4BPMN	Quality for BPMN
QoS	Quality of Service
OSS	Open Source Software
PTC	Project Technical Committee
PSDL	Publish/Subscribe Description Language
SL	Scientific Leader
SE	Service Engine
TDD	Test Driven Development
TSDL	Tuple Spaces Description Language
V&V	Verification and Validation
ULS	Ultra Large Scale
WP	Work Package
WPL	Work Package Leader
WSDM	Web Service Distributed Management
xDL	Description Language
XSA	eXtensible Service Access
XSB	eXtensible Service Bus
XSC	eXecutable Service Composition
XSD	eXtensible Service Discovery

Table Of Contents

List Of Tables

List Of Figures

List of Listings

1 Executive Summary

On M24 of the CHOReOS project, WP5 delivers the first version of the CHOReOS IDRE (Integrated Development and Runtime Environment). The IDRE specification was already delivered in D5.2 [5] at M12. The IDRE relies on the work carried on in WP2, WP3 and WP4. As for a first version of the IDRE, D5.3.1 only includes the work of WP3 and WP4. A second version of the IDRE will integrate the achievement of WP2 as described in the integration plan and the description of work. As a reminder, WP3 activities are dedicated to the implementation of the runtime middleware for executing both Business and Things services and enacting choreographies. While, WP4 activities focus on the realization of the Governance and V&V Framework. At M18 and M24 we deliver the 1st and 2nd versions of both the CHOReOS middleware and the Governance Framework (D3.2.1 [7], D3.2.2 [8], D4.2.1 [9] and D4.2.2 [9]). In D5.3.1, we report the user manuals of these components and give a general cartography of the IDRE.

1.1. Deliverable Objectives

The main objective of this deliverable is to give the needed information for using the CHOReOS components. This deliverable is dedicated to users with technical skills and is not dedicated to end users. D5.3.1is dedicated to the CHOReOS team as a guideline reference for the developed components. More elaborated end user manuals are planned for the Year 3 as the CHOReOS Open Source Software community is active and the software stability is reached. D5.3.1 aims at identifying and listing the IDRE components and sketching its cartography. Based on the information collected among the CHOReOS team, an integrated schema of the CHOReOS scenario is also drawn. Furthermore, a list of the different technologies that are used within the IDRE is made. D5.3.1 gives the steps to follow for using each component of the IDRE. Finally, we also present some supporting tools that are useful for the development of the CHOReOS IDRE.

1.2. Addressing the review recommendations

During the previous review, WP5 was positively evaluated (M12). The recommendations focused on the clarification of the scalability dimensions and an estimation of the size of the deployment infrastructure in terms of machines, nodes, etc. Other recommendations concerning the test bed establishing were also given. Relying on the integration of the works carried on within WP2, WP3 and WP4, WP5 benefits from the clarification of the ultra large scale dimensions realized in the cited work-packages. We refer to the deliverable D2.2 [6], D3.2.2 [8] and D4.2.2 [9]. Moreover, we have initiated the setting of the test bed environment at several levels, i.e. the development of unitary tests for assessing the behavior of the developed components as well as the handling of virtual machines for the middleware and cloud infrastructure. We refer to the deliverable [**?**] and D3.2.2 [8]. During the last year of the project, extensibility tests will be operated on top of the CHOReOS IDRE for evaluating its response to the clarified ULS dimensions.

1.3. Reading Key

We consider a harmonized way of presenting each component. First, a short description of the component and its functionality is given. Then, the installation steps are described considering a unique template (See Listing 1.1).

```
   CHOReOS Install File Template
Component and Subsystem Name - INSTALLATION GUIDE - MM/DD/YYYY
Written by Author(s) Name(s) and Surname(s) (Partner (s))
Authors mails (the person(s) to be contacted if any problem)

== WHAT IS IT? ==
Description and expected Functionality

== REQUIREMENTS ==
Needed Perquisites.

== CONFIGURING IT ==
Needed configurations.

== COMPILING IT ==
Command line to compile the component.

== RUNNING IT ==
Needed steps for running it (script, command line, etc.)

== VERIFYING IF IT WORKS ==
Expected behavior (displaying console, text, etc.)
```

Listing 1.1: CHOReOS Install File Template

1.4. Deliverable Outline

This deliverable is structured as follows, in chapter 2 we present the cartography of the Integrated Development and Runtime Environment is sketched and briefly described. We also include an integrated scenario illustrating the several IDRE Components coming into play in a CHOReOS use case. Second, we dedicate the third chapter to the detail of the IDRE Components user manuals. Third, we describe the tools that supports the CHOReOS Team for designing, developing and testing their code. Finally, chapter 5 concludes the deliverable by giving the components locations in the CHOReOS OW2 Forge, as well as the incoming steps to consider.

2 CHOReOS IDRE Cartography

In this chapter, we describe the CHOReOS IDRE cartography. First in section 2.1, we briefly remind the functionality of the main CHOReOS subsystems. Second, in section 2.2 we sketch an integrated scenario involving all the components. Third, in section 2.3 we list the main baseline and third-party technologies that are used in the project.

2.1. IDRE Overview

The CHOReOS IDRE gathers the most important achievements of the different technological work packages (WP2-3-4). At M24 of the CHOReOS project, the IDRE includes only WP3 and WP4 components. Starting from M24, components from WP2 will be integrated to the IDRE and will be available in the 2nd version of the CHOReOS IDRE at M30. We show the overall cartography of the CHOReOS IDRE at M24 in Figure 2.1. It is composed of the following subsystems and their respective functionality:

- The **CHOREOS Middleware** detailed in D3.2.2 [8] and composed of the eXtensible Service Access (XSA), eXecutable Service Composition (XSC), eXtensible Service Discovery (XSD), and Cloud & Grid Middleware :
  - The **XSA** represents the main runtime infrastructure for accessing business and things services. At M24, the CHOReOS XSA is composed of the *Easy Enterprise Service Bus* (EasyESB) dedicated to the Business Services integration and the choreography enactment (XSC and Component-CD). Moreover, it includes the *Extensible Service Bus* specific binding components, as well as the *Sensor Access Middleware* for the Internet of Things. Finally, the XSA involves an implementation for *Phone Proxy Services* ensuring the communication with device hosted services.
  - The **XSC** is the backbone implementation of the choreography of business services. It relies on the *Component-CoordinationDelegates*, the *Things Composition & Estimation* components.
  - The **XSD** is the service discovery middleware that enables the discovery of heterogeneous services and things. It is built on a plugin technology able to populate a common repository with heterogeneous sources of services. In the first version of the IDRE, it is composed of the *Discovery Plugin Framework*, the *Abstraction-oriented Service Base Management*, and the *Things Discovery Protocol*.
  - The **Cloud & Grid** enables the deployment of services on top of a powerful cloud infrastructure. It is composed of the *Node Pool Manager* that allocates the CHOReOS nodes (virtual machines with some CHOReOS components) in a Cloud, the *Storage Factory* that instantiates a database on a CHOReOS node, the *Service Deployer* that deploys services on a CHOReOS node, the *Enactment Engine* that enacts a given choreography, and the *Grid* that provides access to a grid/cluster computing infrastructure. In CHOReOS, we support the following cloud technologies, Amazon Web Services (AWS EC2) and OpenStack.

- **CHOReOS Governance and V&V Framework** detailed in D4.2.2 [9] is composed of the CHOReOS Governance and V&V (Verification and Validation) components and the monitoring infrastructure.

  - The **CHOReOS Governance and V&V components** provides abilities for governing the resource life cycle. It is composed of the *SLA and Lifecycle Management* that manages and enables the discovery of resources such as services, sla, and choreographies, the *Partes* framework that operates testing of services and participants of a choreography, the *ServicePot* framework that provides choreography testing and discovery, the *CRank* that realizes services ranking according to their testing, and the *Rehearsal* that performs services testing and V&V .
  - The **CHOReOS Monitoring Infrastructure** offers a multi source monitoring of running services and cloud resources. It is composed of the *Glimpse* tool that provides Complex Event Processing, the *Business Service Monitoring (BSM)* that monitors business services running on top of the CHOReOS XSA, and the *Platform Monitoring* that controls the cloud resource availability and triggers its adaptation.

  Both the CHOReOS Middleware and the governance framework are assessed thanks to the choreographies and services implemented within the Passenger Friendly Airport (WP6), the Dynaroute (WP8) and ongoing experiments on the ACRB use case (WP7).

The IDRE is implemented using the Java language and its source code is available in the CHOReOS OW2 forge repository. The release of the open source packages is planned for M30. The code is public and open source, it can be downloaded from the OW2 with the following command:

```
1 svn checkout svn+ssh://developername@svn.forge.objectweb.org/svnroot/choreos/trunk
```

Committing to the forge is possible is possible, providing that the developer account is created and the credentials are granted.

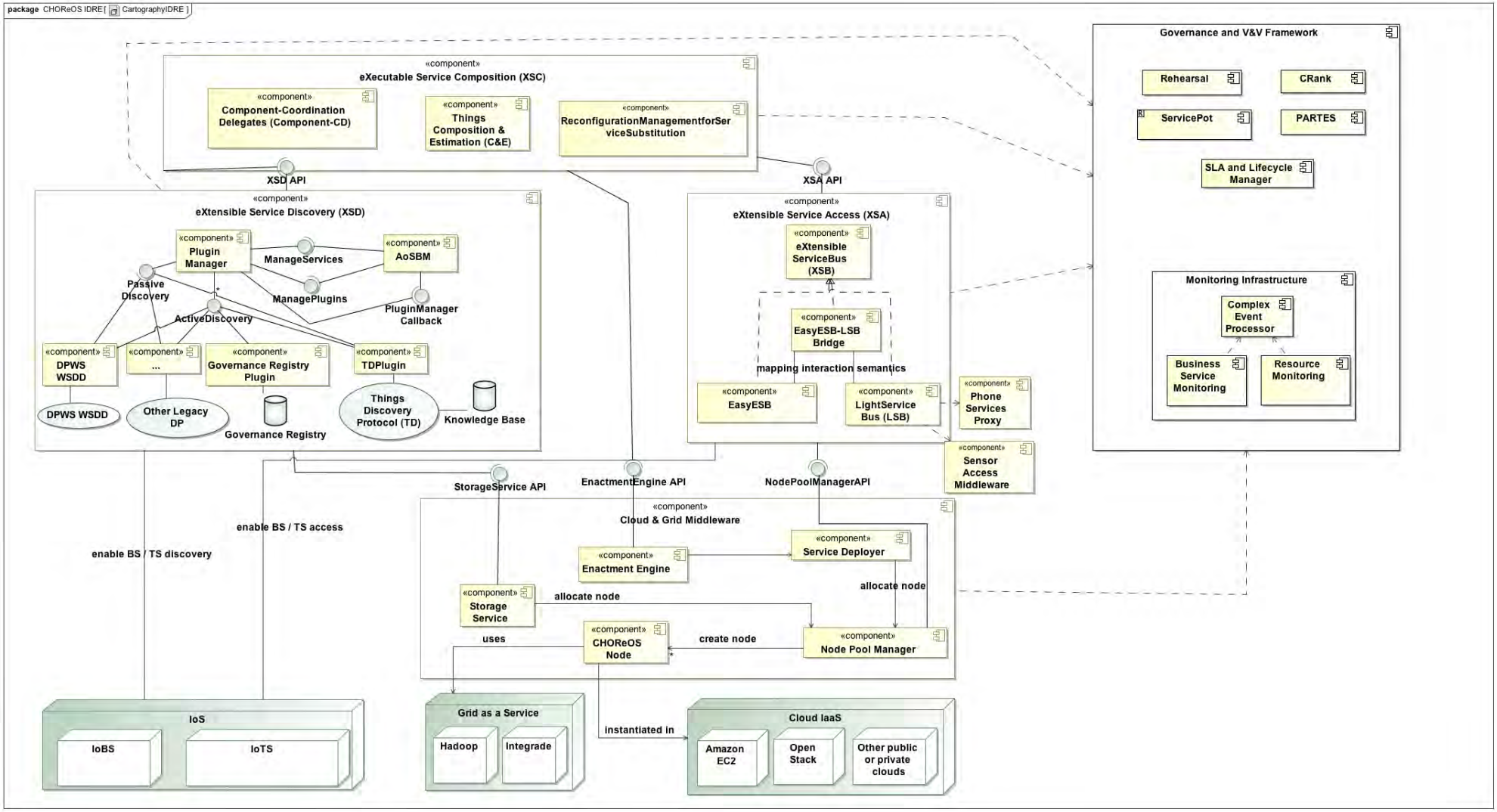

Figure 2.1: Overall architecture of the CHOReOS IDRE at M24

2.2. CHOReOS Integrated Scenario

Within the CHOReOS IDRE, the subsystems as well as their components get involved into a common integrated scenario as follows. This section aims at clarifying the process that happens between the IDRE Components. In Figure 2.2, we illustrate this process. At each step, we refer to the work package, the subsystem responsible for achieving the task.

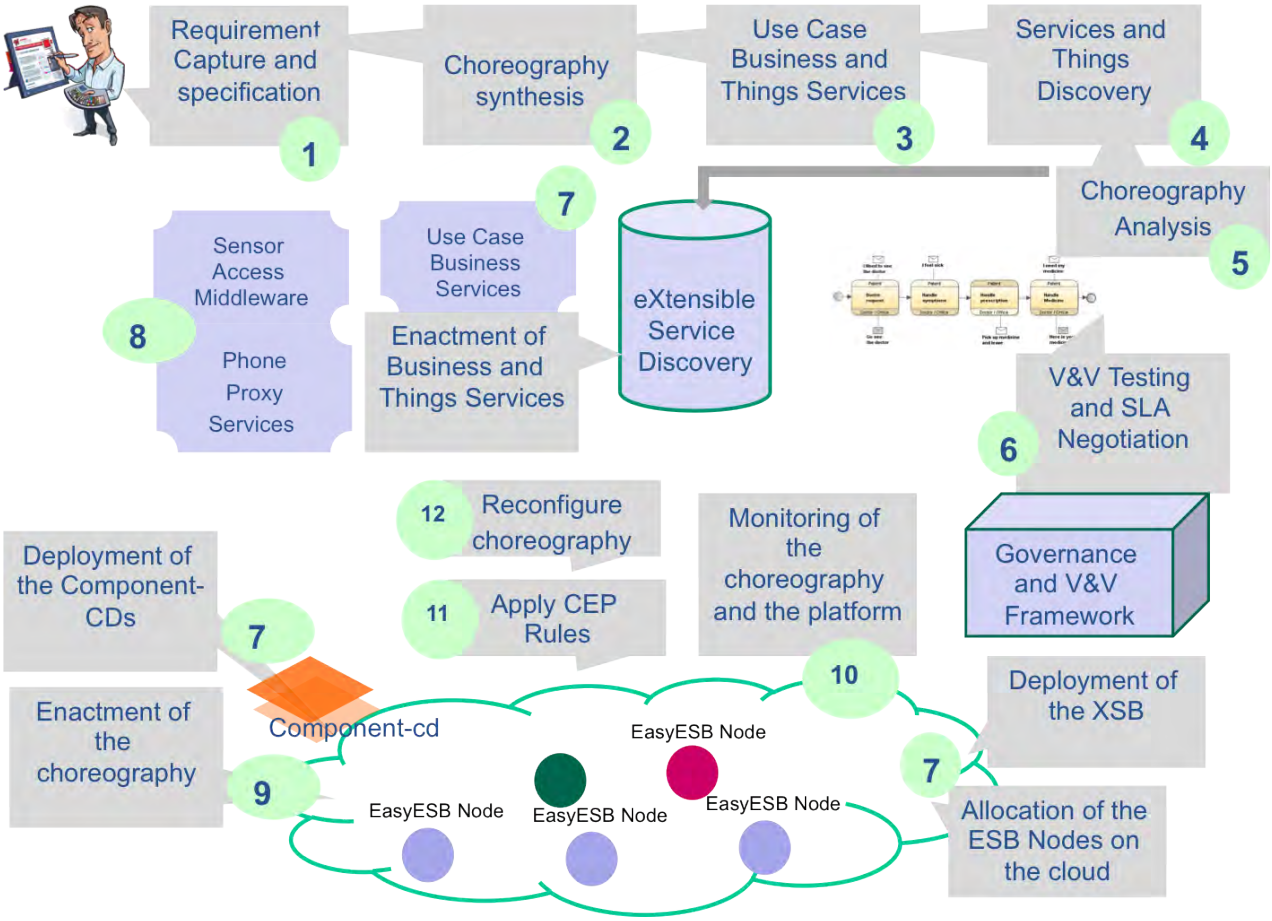

Figure 2.2: CHOReOS Integrated Deployment Scenario

1) The user requirements are captured thanks to the tools developed within WP2 [CHOReOS Development Environment-**Requirements Specification Tools**-WP2]

2) The choreography synthesis takes place and produces a coordination logic considering the gathered requirements and the discovered services. [CHOReOS Development Environment-**Synthesis Processor**-WP2]

3) A preliminary task consists in implementing the use case services of both Business Services and Things Services [WP6-7-8]

4) Then, thanks to the eXtensible Service Discovery middleware, the services Publication and Lookup are enabled [Extensible Service Discovery-**Abstraction-Oriented Service Base Management**-WP3]

5) Another preliminary task that is realized at development time consists in analyzing the choreography with regard to the scalability dimensions [CHOReOS Development Environment-**Choreography Analyser**-WP2].

6) As for the choreographies, the services need to be tested, governed and the service level agreements need to be negotiated for further assessment [Governance and V&V Framework- **Rehearsal, ServicePot, SLA & Lifecycle Manager**- WP4].

7) The next step consists in enacting the Business services [Cloud & Grid Middleware-**Enactment Engine**-WP3] and Allocating the ESB nodes on the cloud [Extensible Service Access-**EasyESB**-WP3]. The EasyESB nodes deployed on the cloud nodes embed the Component Coordination Delegates with the synthesized choreography [Executable Service Composition-**Component-CD**-WP3]. This step is made in parallel with the step 8 and the deployment of the XSB and the corresponding SU binding components [Extensible Service Access- **Extensible Service Bus**-WP3].

8) Furthermore, the Things-based services are enacted [Extensible Service Access- Light Service Bus- **Sensor Access Middleware and the Phone Proxy Services**-WP3].

9) Once the business and things-based services participating to the choreography are running, the choreography can be enacted. [Extensible Service Access-Extensible Service Composition-Cloud and Grid Middleware-WP3]

10) It is possible then to monitor the choreography and the cloud resources thanks to the CHOReOS multi-source monitoring framework [CHOReOS monitoring framework- **Business Service Monitor** - **Resource Monitoring**-WP4]

11) The generated events are interpreted by the Complex Event Processor (CEP) for triggering the right rules [CHOReOS Monitoring Framework- **Glimpse**-WP4]

12) The CEP can trigger an event for the reconfiguration of the choreography [Executable Service Composition-**Reconfiguration and management for service Distribution**-WP3]

2.3. Technologies

In Table 2.1, we list the baseline (B) and third-party (T) technologies that are used within the CHOReOS IDRE. This list is meant to be extended as the needs of the project are evolving.

CHOReOS IDRE Contribution	Technology
Choreography Specification	Business Process Management Notation (BPMN2.0) (B)
Service Description	Web Service Description Language (WSDL1.1) (B)
Service Level Agreements	Web Services Level Agreement (WSLA) (B)
Event Notification	WS Notification (B)
Services Non functional Properties	Web Service Distribution Management (WSDM) (B)
Business Code	Java 2 Enterprise Edition (J2EE) (B)
Service Registry Protocol	Universal Description Discovery and Integration (UDDI) (B)
Things Discovery Protocol	Devices Profile for Web Services (DPWS) (B)
Cloud API	Amazon Web Services EC2, OpenStack (T)
Sensor Access Middleware	Java for Android Phones (B)
Phone Proxy Services	Rest Web Services (B)
Services Query	Service Base Query Language (B)
Platform Monitoring	Ganglia monitoring system (T)

Table 2.1: Used Technologies within the CHOReOS IDRE

3 CHOReOS IDRE User Manual

This chapter is dedicated to the description of the user manuals of each component involved in the CHOReOS IDRE. We believe this step is necessary to reach a general comprehension of the project results and to identify the integration points and scenarios that will bring into play all the components. This first version of the IDRE is a vision of the components involved in the middleware and the governance framework. They are still under development and improvements and are still hosted in the trunk of the CHOReOS forge. It's noteworthy to mention that the user manual delivered with this version are dedicated to technical users and to the CHOReOS team developers. A more stable version of the software accompanied with elaborated end user manuals is to be released by M30 of the project as Open Source Software. Hereafter, in Table 3.1 we list the CHOReOS components of this first version.

Subsystem	Component	Repository Location
XSC	Component-cd	choreos/trunk/executable_service_composition/dsb_cd_implementation
	C&E	choreos/trunk/executable-service-composition/c-and-e/
XSA	XSB	choreos/trunk/extensible-service-access/xsb-over-dsb
	Sensor Access Middleware	choreos/trunk/extensible-service-access/lsb/sensorAccess
	Phone Proxy Services	choreos/trunk/extensible-service-access/lsb/PhoneServicesProxy/
	EasyESB	http://research.petalslink.org/display/easyesb/Downloads
XSD	AOSMB	choreos/trunk/extensible_service_discovery/AoSBM
	Discovery Plugin Framework	choreos/trunk/extensible_service_discovery/plugin_manager
	Things Discovery	choreos/trunk/extensible_service_discovery/
Cloud and Grid	Node Pool Manager	choreos/trunk/cloud/NodePoolManager
	Storage Factory	choreos/trunk/cloud/ServiceFactory
	Service Deployer	choreos/trunk/cloud/ServiceDeployer
	Grid	choreos/trunk/grid/integrade
CHOReOS Development and Runtime Governance	Rehearsal	governance/tdd/rehearsal
	Service Pot	governance/servicepot
	Partes	governance/v_and_v/partes
	CRank	governance/v_and_v/crank
	SLA&LifecycleManagement	governance/sla_and_lifecycle_manager
CHOReOS Monitoring Framework	Glimpse	governance/component-glimpse
	Business Service Monitoring	governance/new-bsm-distribution-choreos
	Platform Monitoring	cloud/Monitoring

Table 3.1: CHOReOS IDRE Components Availability

CHOReOS Middleware User Manuals

The reference implementation of the CHOReOS Middleware is available in [8].

3.1. eXecutable Service Composition

3.1.1. Component-CD User Manual

The Component Coordination Delegates (Component-CD) is dedicated to the choreographies enactment. It takes as input the choreography synthesized during WP2 activities. It encapsulates a coordination logic (available at M24). Right now, it takes as input the configuration files takes as input 2 configuration files listing the services invocation addresses and their descriptions. The Component-CD is dedicated to be run in the EasyESB middleware as a bus component (See Sec.3.2.4).

Install Guidelines

```
   CHOReOS Install Files Template
Component-CD and XSC - INSTALLATION GUIDE - 10/01/2012

Written by Julien Lesbegueries (Linagora)
nicolas.salatge@linagora.com
julien.lesbegueries@linagora.com
amira.ben-hamida@linagora.com

== WHAT IS IT? ==
The component-cd is the implementation of the coordination delegates.
It is adapted to the EasyESB Middleware. It is deployed and started as a component within the
    CHOReOS XSA Middleware. The Component-CD takes as input 2 configuration files listing the
    services invocation addresses and their descriptions.

== REQUIREMENTS ==
A CHOReOS EasyESB Distribution is required. Refer to Section XSA/EasyESB.

== CONFIGURING IT ==
 1.EasyESB and its SDK can be downloaded from this
 URL http://research.petalslink.org/display/easyesb/Downloads.
 2.Download the latest release of EasyESB and install it:
  On Windows OS, click on installer
  On Linux OS, java -jar EasyESB-installer-VERSION.jar
3.The configuration files informing about the services invocation addresses and the services
    descriptions need to be prepared.

== COMPILING IT ==
You need to compile the EasyESB distribution using this command:
  mvn clean install -Pdistrib

== RUNNING IT ==
After the compilation step, Go to the bin directory of project home (shortcut for users of the
    Windows OS):
 1.On Windows OS, start: ./startup.bat
 2.On Linux OS, start: ./startup.sh
 (remember to give write access on this file: chmod a+x startup.sh)

== VERIFYING IF IT WORKS ==
Once started successfully, the ESB client enables the administration of the bus nodes (can be used
     in scripts)
In the console mode, you need to type:
1.c <administration_address_of_ESB_node> to connect to a node (for instance http://localhost:8180/
    services/adminExternalEndpoint), then,
2.d <cd_conf_file> <cd_wsdl_file> to deploy a CD (the cd_conf_file must is a java properties file
    for now, containing at least a endpoint key and
endpoint value of real service to invoke. For instance: endpoint=AirlineService)
To display the help text, type 'h'.
```

Listing 3.1: Component Coordination Delegates User Manual

3.1.2. Composition & Estimation User Manual

The C&E component is responsible for the composition of things-based services. Those services should be provided by devices already registered with the Things Registration Manager and discovered by the Things Query Manager, both of which are parts of the Things Discovery Component. The C&E accesses the Things Discovery component to find real world services that provide the needed measurements, after which it retrieves their data through the Service Access component. In the following, we present the installation steps and requirements for the component to function properly.

Install Guidelines

```
Composition Manager - INSTALLATION GUIDE - 29/09/2012
Written by Hachem Sara - Inria
mail: sara.hachem@inria.fr

==WHAT IS IT?==

The Composition Manager should be accessed to generate thing-based service compositions defined as
     mathematical formulas in the SensorsAndActuators RDF file.
It is part of the Executable Service Composition middleware and CHOReOS middleware. The component
    can be downloaded from Maven as a jar file. The source code is available at:
"http://forge.ow2.org/projects/choreos/" and "/trunk/executable-service-composition/c-and-e/"

==REQUIREMENTS==

#1 Java 6
#2 Apache Maven
The Composition Manager uses three libraries and a Mathematica file:

- jUnit-4.7:
Needed for jUnit tests. can be found on maven repositories

- jena-2.6.4:
Needed to handle ontologies and RDF files.
Source: http://sourceforge.net/projects/jena/files/Jena/Jena-2.6.4/
Can be found on maven repositories

- IoTCommonObjects V1.4
Needed for service access, sensor data description, common classes. Deployed on: "http://maven.
    petalslink.com".
Source: committed the project to the OW2 repository in Executable-Service-Discovery

- fit.m file
The file is used by Mathematica to compute some needed values. The file has has been added to the
    top directory and
the path in the code points to its location there.
Path: ./fit.m

==CONFIGURING IT==

The component uses a RESTful service, provided in choreos/trunk/extensible-service-discovery/
    registry_manager.
 and it assumes that the service has been deployed on the the following address:
 "http://things.vtrip.net:8080/RegistryManager-2.0/rest/"

It also uses Mathematica software to compute some values, the path of the software can be passed
    as an argument if a default
path is not specified in the code. the property to set is "org.ow2.choreos.mathematicaAppDirPath".
e.g. -Dorg.ow2.choreos.mathematicaAppDirPath="/Applications/Mathematica.app/Contents/MacOS/"
If not path is provided the process will stop.
The rdf and mathematica script files should be placed in the top directory or their paths should
be specified based on the following properties:
The property to set for the Mathematica script file is "org.ow2.choreos.scriptDirPath"
The property to set for the RDF file is "org.ow2.choreos.registration.rdfFilePath"

==VERIFYING IF IT WORKS==
Several test classes have been added to /test/java under the src folder in the project.
```

Listing 3.2: C&E User Manual

3.2. eXtensible Service Access

3.2.1. XSB User Manual

The eXtensible Service Bus (XSB) provides support for the seamless integration of heterogeneous interaction paradigms, more specifically client/server (CS), publish/subscribe (PS) and tuplespace (TS). This is performed by extending the EasyESB Enterprise Service Bus with the Generic Application (GA) protocol on top of the underlying ESB protocol. XSB Binding Components (BCs) are designed to be easily extensible to support new middleware platforms or new interaction paradigms.

Install Guidelines

```
XSB DPWS Binding Component over EasyESB - INSTALLATION GUIDE - 17/09/2012
Written by Sandrine Beauche (Inria)
sandrine.beauche@inria.fr

== WHAT IS IT? ==
This module allows to execute the XSB DPWS Binding Component with EasyESB.

== REQUIREMENTS ==

Before you compile the project you need to install
#1 Java 6
#2 Maven 3 (http://maven.apache.org/download.html)

== CONFIGURING IT ==
== COMPILING IT ==
1 download sources: trunk/extensible-service-access/xsb-over-dsb/connectors/clientServer/
    WebServices/DPWSBindingComponent/easyesbjmeds/

2. Add this repository to your settings.xml

<repository>
  <id>choreos-petalsLink</id>
  <name>choreos maven repository</name>
  <url>http://maven.petalslink.com/repo</url>
</repository>

3. execute mvn install to obtain the binding, or mvn install -Pdistrib to obtain a easyESB
    distribution with the DPWS
Binding Component.

== RUNNING IT ==

run target/easyesbjmeds/bin/startup.bat to start the easyESB distribution with the DPWS
Binding Component.

== VERIFYING IF IT WORKS ==

You can verify the binding component by running 2 scenarios
-> 2 DPWS applications have a one-way exchange through the XSB Binding Components. The first
    application sends
a notification that is received by the second application as a one-way invocation.
-> 2 DPWS applications have a two-way exchange through the XSB Binding Components. The first
    application sends
a solicit-response that is received by the second application as a two-way invocation.

Running the first scenario:
1) start the application that receive the one-way invocation: java -jar Easyesbjmeds-1.0-SNAPSHOT-
    OneWaySystem.jar
2) In the menu, choose option 1 to start the system
3) start a easy ESB node with the DPWS Binding Component
4) deploy the corresponding service unit: src/test/resources/oneWay/UnitOneWay.xml.
    DPWSDefinitionOneWay.xml should be in the
same directory as it is referenced by the xml deployment file.
5) start a second easy ESB node with a DPWS Binding Component
6) deploy the service unit of the system that sends the notification: src/test/resources/
    notification/UnitNotification.xml.
```

```
7) start the application that sends the notification: java -jar Easyesbjmeds-1.0-SNAPSHOT-
    NotificationSystem.jar
8) choose option 1 in the menu to start the application
9) choose option 2 and argument 1 to send a notification

10) stop the client application by choosing option 1 and then 2
11) stop the easyESB nodes
12) stop the server application by choosing option 1 and then 2



Running the second scenario:
1) start the application that receive the two-way invocation: java -jar Easyesbjmeds-1.0-SNAPSHOT-
    TwoWaySystem.jar
2) In the menu, choose option 1 to start the system
3) start a easy ESB node with the DPWS Binding Component
4) deploy the corresponding service unit: src/test/resources/invoke/UnitTwoWay.xml.
    DPWSDefinitionTwoWay.xml should be in the
same directory as it is referenced by the xml deployment file.
5) start a second easy ESB node with a DPWS Binding Component
6) deploy the service unit of the system that sends the notification: src/test/resources/solicit/
    UnitSolicit.xml.
7) start the application that sends the notification: java -jar Easyesbjmeds-1.0-SNAPSHOT-
    SolicitSystem.jar
8) choose option 1 in the menu to start the application
9) choose option 2 and argument 1 to send a notification

10) stop the client application by choosing option 1 and then 2
11) stop the easyESB nodes
12) stop the server application by choosing option 1 and then 2

++ source code examples
== AUTOMATED TESTS ==

The preceding scenarios are implemented with integration tests with JUnit. So
remove skipping test configuration for surefire and then
execute mvn test to launch them in a automated testing.
```

Listing 3.3: XSB DPWS Binding Component with EasyESB User Manual

```
XSB JavaSpaces Binding Component over EasyESB - INSTALLATION GUIDE - 17/09/2012
Written by Sandrine Beauche (Inria)
sandrine.beauche@inria.fr

== WHAT IS IT? ==
This module allows to execute the XSB JavaSpaces Binding Component with EasyESB.

== REQUIREMENTS ==

Before you compile the project you need to install
#1 Java 6
#2 Maven 3 (http://maven.apache.org/download.html)
#3 Rio (http://www.rio-project.org/)
    -> download and unzip the rio tarball
   -> set RIO_HOME to the home

== CONFIGURING IT ==
== COMPILING IT ==
1 download sources: trunk/extensible-service-access/xsb-over-dsb/connectors/tupleSpaces/JavaSpaces
    /JSBindingComponent/easyesbjini/

2. Add this repository to your settings.xml

<repository>
  <id>choreos-petalsLink</id>
  <name>choreos maven repository</name>
  <url>http://maven.petalslink.com/repo</url>
</repository>

3. execute mvn install to obtain the binding, or mvn install -Pdistrib to obtain a easyESB
    distribution with the JavaSpaces
Binding Component.
```

```
== RUNNING IT ==

run target/easyesbjmeds/bin/startup.bat to start the easyESB distribution with the JavaSpaces
Binding Component.

== VERIFYING IF IT WORKS ==

You can verify the binding component by running 2 scenarios
-> 2 JavaSpaces applications have a one-way exchange through the XSB Binding Components. The first
     application sends
a notification that is received by the second application as a one-way invocation.


Running the first scenario:
1) start the application that receive the one-way invocation: java -jar Easyesbjmeds-1.0-SNAPSHOT-
    OutSystem.jar
2) In the menu, choose option 1 to start the system
3) start a easy ESB node with the JavaSpaces Binding Component
4) deploy the corresponding service unit: src/test/resources/out/UnitOut.xml. JMSDefinitionOut.xml
     should be in the
same directory as it is referenced by the xml deployment file.
5) start a second easy ESB node with a JavaSpaces Binding Component
6) deploy the service unit of the system that sends the notification: src/test/resources/register/
    UnitRegister.xml.
7) start the application that sends the notification: java -jar Easyesbjmeds-1.0-SNAPSHOT-
    RegisterSystem.jar
8) choose option 1 in the menu to start the application
9) choose option 2 and argument 1 to send a notification

10) stop the client application by choosing option 1 and then 2
11) stop the easyESB nodes
12) stop the server application by choosing option 1 and then 2

++ source code examples
== AUTOMATED TESTS ==

The preceding scenarios are implemented with integration tests with JUnit. So
remove skipping test configuration for surefire and then
execute mvn test to launch them in a automated testing.
```

Listing 3.4: XSB JavaSpaces Binding Component with EasyESB User Manual

```
XSB JMS Binding Component over EasyESB - INSTALLATION GUIDE - 17/09/2012
Written by Sandrine Beauche (Inria)
sandrine.beauche@inria.fr

== WHAT IS IT? ==
This module allows to execute the XSB JMS Binding Component with EasyESB.

== REQUIREMENTS ==

Before you compile the project you need to install
#1 Java 6
#2 Maven 3 (http://maven.apache.org/download.html)

== COMPILING IT ==
1 download sources: trunk/extensible-service-access/xsb-over-dsb/connectors/publishSubscribe/JMS/
    JMSBindingComponent/easyesbjoram/

2. Add this repository to your settings.xml

<repository>
  <id>choreos-petalsLink</id>
  <name>choreos maven repository</name>
  <url>http://maven.petalslink.com/repo</url>
</repository>

3. execute mvn install to obtain the binding, or mvn install -Pdistrib to obtain a easyESB
    distribution with the JMS
Binding Component.
```

```
== RUNNING IT ==

run target/easyesbjmeds/bin/startup.bat to start the easyESB distribution with the JMS
Binding Component.

== VERIFYING IF IT WORKS ==

You can verify the binding component by running 2 scenarios
-> 2 JMS applications have a one-way exchange through the XSB Binding Components. The first
    application sends
a notification that is received by the second application as a one-way invocation.


Running the first scenario:
1) start the application that receive the one-way invocation: java -jar Easyesbjmeds-1.0-SNAPSHOT-
    PublishSystem.jar
2) In the menu, choose option 1 to start the system
3) start a easy ESB node with the JMS Binding Component
4) deploy the corresponding service unit: src/test/resources/publish/UnitPublish.xml.
    JMSDefinitionPublish.xml should be in the
same directory as it is referenced by the xml deployment file.
5) start a second easy ESB node with a JMS Binding Component
6) deploy the service unit of the system that sends the notification: src/test/resources/subscribe
    /UnitSubscribe.xml.
7) start the application that sends the notification: java -jar Easyesbjmeds-1.0-SNAPSHOT-
    SubscribeSystem.jar
8) choose option 1 in the menu to start the application
9) choose option 2 and argument 1 to send a notification

10) stop the client application by choosing option 1 and then 2
11) stop the easyESB nodes
12) stop the server application by choosing option 1 and then 2

== AUTOMATED TESTS ==

The preceding scenarios are implemented with integration tests with JUnit. So
remove skipping test configuration for surefire and then
execute mvn test to launch them in a automated testing.
```

Listing 3.5: XSB JMS Binding Component with EasyESB

3.2.2. Sensor Access Middleware User Manual

The Sensor Access Middleware enables high-level programming on top of smartphone sensors by providing: (i) a platform-independent API which separates application logic from sensor logic and wraps all types of sensors and their data into a common representation; and (ii) support for easily adding drivers for new sensors that can transparently bind with the API and provide access to sensed data . The Sensor Access Middleware has been implemented in Java for Android phones.

Install Guidelines

```
Sensor Access Middleware - INSTALLATION GUIDE - 04/04/12

Written by Kostas Lekkas, Thanassis Parathyras (VTRIP)

lekkas@vtrip.net
aparathyras@vtrip.net

== WHAT IS IT? ==

The AndroidSensorAccess Android application enables access to native sensor data through a RESTful
API. The native sensor data is extracted using a mobile-phone-level middleware provided by INRIA.
For more information on the architecture of the native sensor access middleware please consider
reading D3.2.1.

The aforementioned RESTful API is currently being implemented using the Restlet framework
(http://www.restlet.org/) .

== REQUIREMENTS ==

In order to install the application, you need to download and install the latest Android
Development Kit. You can find detailed instructions on how to do it here:
http://developer.android.com/sdk/index.html

== CONFIGURING IT ==

The application does not need any special configuration. Just build, install and run it on a
device or emulator.

== COMPILING IT ==

Provided that you have installed the latest Android SDK, right click on the 'AndroidSensorAccess'
project and choose 'Build project'

== RUNNING IT ==

Provided that you have installed the latest Android SDK, just click 'Run' . Even if you have not
previously built the project, the SDK will do it automatically for you.

== VERIFYING IF IT WORKS ==

After you have installed and started the application, tap 'Enable'. Then, from a host that is on
the same network with the mobile phone, issue the following REST request using 'curl' :

curl -i -X GET http://{mobile-phone-ip}:8182/getnoiselevel

If everything works correctly, you should get an HTTP response with status 200 and a similar
body:

{
  "sensorDataResponse":{
    "timestamp":"0",
    "value":"-66.191",
    "dataType":"org.ow2.choreos.sensordata.double.noiselevel"
  }
}
```

Listing 3.6: Sensor Access Middleware User Manual

3.2.3. Phone Proxy Services User Manual

The Phone Services Proxy provides indirect access to services that reside on mobile devices. It was introduced as a solution that deals with a number of issues related to mobile communication, such as the use of NAT, transient IP addresses and temporary disconnections. Clients (e.g. Things Query Manager) send their requests and receive the corresponding results via the Phone Services Proxy, rather than directly to/from the devices themselves. Mobile devices with Sensor Access Middleware poll the Phone Services Proxy for client requests that have been issued towards them and in turn send replies that will be forwarded to the respective clients. All communication interfaces are implemented using RESTful web services.

Install Guidelines

```
Phone Services Proxy - INSTALLATION GUIDE - 10/09/2012

Written by Kostas Lekkas, Thanassis Parathyras (VTRIP)

lekkas@vtrip.net
aparathyras@vtrip.net

== WHAT IS IT? ==

The Phone Services Proxy component was introduced in the M24 release of the CHOReOS middleware,
as a solution that deals with a number of issues related to mobile communication, such as
the use of NAT, transient IP addesses, enforcement of asymmetrical communication and temporary
disconnections. Clients that wish to communicate with services residing on mobile devices (e.g.
the Things Query Manager) send their requests and receive the corresponding results via the
Phone Services Proxy (henceforth referred to as proxy), rather than directly to/from the
devices themselves. Mobile devices retrieve, in a pull-based fashion, client requests that have
been issued towards them via theproxy, and in turn send replies to these requests in order for
them to be forwarded to the respective clients. The introduced proxy mechanism as described
here, constitutes the main building block towards the realization of Light Service Bus(LSB),
by providing a solid access solution to sensor data provided at the Things level.

== REQUIREMENTS ==

All library requirements are handled by Maven. You only need an Apache Tomcat web container
(version 6+) to test this component.

== CONFIGURING IT ==
The application needs no special configuration.

== COMPILING IT ==

Enter the root directory of the project "choreos/trunk/extensible-service-access/lsb/
    PhoneServicesProxy/" and run 'mvn clean install'

== RUNNING IT ==

Given that the compilation was successful, deploy the .war on a Tomcat 6 compatible web server.
You can find the .war file in 'target/PhoneServicesProxy-1.0.war'

== VERIFYING IF IT WORKS ==

After you have deployed the .war package on the web server -and assuming that the web server
runs on the local host- issue the following REST request using 'curl' :

curl -X GET http://localhost:8080/PhoneServicesProxy-1.0/pendingrequests/12345abc

If everything works correctly, you should get an HTTP response with status 200 and a similar
body:

{
  "pendingRequests":[]
}
```

Listing 3.7: Phone Service Proxy User Manual

3.2.4. EasyESB User Manual

The EasyESB (Enterprise Service Bus) is part of the eXtensible Service Access, it is responsible for enabling the access to services involved in choreographies. Moreover, it hosts the Component-CD responsible for the enactment of the choreographies, and presented in Sec. 3.1.1. EasyESB is also integrated with the XSB Bindings 3.2.1 and deployed on top of the Cloud infrastructure (See Sec. 3.4). Hereafter is described its user manual.

Install Guidelines

```
   CHOReOS Install Files Template

EasyESB and XSA - INSTALLATION GUIDE - 10/01/2012

Written by Nicolas Salatge (Linagora)

nicolas.salatge@linagora.com
julien.lesbegueries@linagora.com
amira.ben-hamida@linagora.com

== WHAT IS IT? ==
EasyESB is a lightweight services bus based on advanced SOA paradigms.
It is based on MDA (Model Driven Architecture) approaches.
It is part of the extensible Service Access middleware and part of
the CHOReOS Middleware. More documentation is available
here: http://research.petalslink.org/display/easyesb

== REQUIREMENTS ==
  1. Download java jdk 1.6
  2. Set JAVA_HOME in your classpath
  3. Copy this endorsed directory in jre library:
  $JAVA_HOME/jre/lib (for linux/windows) and $JAVA_HOME/lib (for mac)
  4. Download maven 2.2.1
  5. Set MAVEN_HOME in your classpath
  6. Set MAVEN_OPTS=-Xms512m -Xmx1024m -XX:PermSize=256m in your classpath

== CONFIGURING IT ==
 1.EasyESB and its SDK can be downloaded from this
 URL http://research.petalslink.org/display/easyesb/Downloads.
 2.Download the latest release of EasyESB and install it:
  On Windows OS, click on installer
  On Linux OS, java -jar EasyESB-installer-VERSION.jar

== COMPILING IT ==
Not required. The source code is available here:
https://svn.petalslink.org/svnroot/trunk/research/dev/experimental/easyesb
Login: anonymous
Password : anonymous

== RUNNING IT ==
Go to the bin directory of project home (shortcut for users of the Windows OS):
 1.On Windows OS, start: ./startup.bat
 2.On Linux OS, start: ./startup.sh
 (remember to give write access on this file: chmod a+x startup.sh)

== VERIFYING IF IT WORKS ==
When started successfully the console presented in this figure
(EasyESB Started Screenshot) appears.
EasyESB is stopped by hitting the 'X' button of the console.

Once started, EasyESB offers an administration service
(EasyESB Administration Service Screenshot).
By hitting "i" on the command line the EasyESB information is printed.
There will be exposed the administration Web Service URL.
```

Listing 3.8: EasyESB User Manual

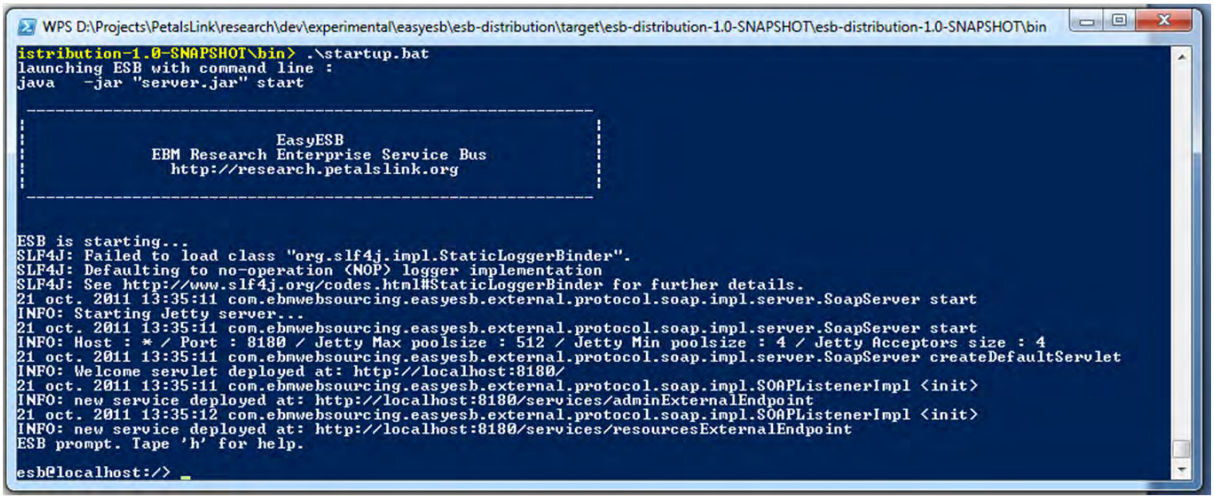

Figure 3.1: EasyESB Started Screenshot

Thanks to this url we can use web service clients such as EasiestDemo Client ([1]) or SoapUI to invoke administration operations (See Section 4.2).

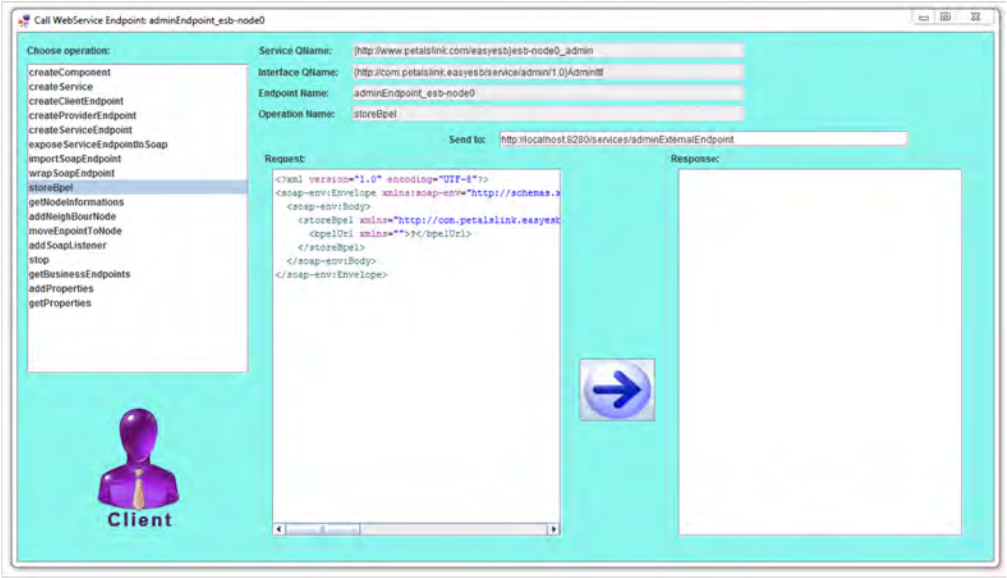

Figure 3.2: EasyESB Administration Service Screenshot

[1] EasiestDemo is a Web service client developed within Linagora R&D to make more user friendly the administration of the bus http://research.petalslink.org/display/easiestdemo/Client

3.3. eXtensible Service Discovery

3.3.1. AOSMB User Manual

The source code of the Abstraction-oriented Service Base Management (AoSBM) is located at `choreos/trunk/extensible_service_discovery/AoSBM`. The AoSBM allows to register collections of service descriptions, organize these service descriptions with respect to functional/non-functional abstractions. Moreover, the AoSBM allows execute service discovery queries, which retrieve abstractions, along with related information that concerns the concrete services that are represented by the retrieved abstractions. At this stage, the first versions of the main AoSBM components have been implemented. Following we provide further installation and use instructions for the AoSBM.

Install Guidelines

```
CHOReOS Install Files Template

AoSBM INSTALLATION GUIDE - 01/10/2012
Written by Dionysis Athanasopoulos (UOI)
Author e-mail (dionysiscsuoi@yahoo.com)


== WHAT IS IT? ==

The AoSBM stands for Abstractions-oriented Service Base Management.
This particular component is used for constructing hierarchies of
functional/non-functional abstractions out of collections of service
descriptions that are registered to the AoSBM. AoSBM consists of 4 main
components:

1) the Service Registration component that allows to construct
collections of service descriptions gathered with the help of
the Plugin Manager.

The service descriptions are parsed and transformed according to
the unified CHOReOS service-oriented component model (defined
in WP1).

2) the Abstraction-driven Service Organization component allows
to perform clustering on a collection of service descriptions
and construct hierarchies of functional/non-functional abstractions.

3) The Service Base Component stores/retrieves service descriptions
and abstractions in/to a database (defined in WP2).

4) The Query Engine component allows to query the Service Base for
abstractions based on a specific language called SBQL (defined in
WP2).

The input to the Query Engine is a file written in SBQL. The result
of the query is a set of abstractions which meet the query's
specification.

== REQUIREMENTS ==
Before you compile the project you need to install:
#1 Java v1.7
#2 Maven 3 (http://maven.apache.org/download.html)
#3 MySQL v5.1

== CONFIGURING IT ==

After installing Java and MySql, the 'mysql connector' for Java should
be added in the MySQL installation. The 'mysql connector' can be
downloaded from http://www.mysql.com/downloads/connector/j/).

After downloading the 'mysql connector', you extract the .zip file and
the extracted folder (e.g. mysql-connector-java-5.1.19) must be copied
into the MySql directory (default C:\Program Files\MySQL).
```

```
In addition, the file mysql-connector-java-5.1.19-bin.jar contained
in the extracted folder must be copied to the Java sub-directories
xxxx\jre6\lib\ext\ and xxxx\jdk1.6.0_11\jre\lib\ext where xxxx is the
Java home directory (default C:\Program Files\Java).

Before using the Service Base and Query Engine components, the
database tables must be created. To create them, you should run the
CreateTables.sql script.

Finally, the system variable CLASSPATH must be updated properly, so
that it includes the JDBC driver. The JDBC driver is under the MySql
installation directory
(default C:\Program Files\MySQL\mysql-connector-java-5.1.19\src).

== COMPILING IT ==
The AoSBM can be easily compiled via the Eclipse IDE.

== LAUNCHING IT ==

The class 'GUI/AoSBM.java' contains the 'main' method of the AoSBM
code that launches the AoSBM GUI. Though the GUI you can register
service collections, organize the collections with respect to
functional/non-functional abstractions, store and query the
abstractions and query easily as described in the short user
guide that follows.

== VERIFYING IF IT WORKS ==
The AoSBM can be tried out with the service collections that are
given in 'Resources/Source(WSDLv1.1)'.

== AUTOMATED TESTS ==
Simple automated unit tests for the AoSBM are further provided
in 'src/test'.
```

Listing 3.9: Abstraction Oriented Service Base Management

AoSBM Usage

This component is dedicated to the end users for the registration of service and thus it provides a Graphical User Interface (GUI). The AoSBM GUI consists of 3 main tabs that allow to register service collections, organize the collections with respect to hierarchies of functional/non-functional abstractions and query the hierarchies for abstractions that meet requirements specified in SBQL. Figure 3.3 shows the `Service Registration` tab. In this tab you can create a new collection of services by using the `Select Source` button. Using the file browser that is launched after pushing this button, you have to select a folder that contains WSDLv1.1 service descriptions and QoS descriptions (sample service descriptions can be found in `Resources/Source(WSDLv1.1)`). After parsing the input files (wsdl, xml, QoS, etc.) you can store in the `Service Base` the produced information, by clicking on the `Store Collection` button.

Figure 3.4 shows the `Abstraction-oriented Service Organization` tab. In this tab click on a previously created collection of services (e.g., `Collection 1`). Then, select the organization mode (`Functional and non-functional are independently developed` if the mode that is currently supported). Finally, press the button `Organize Services`.

In the newly opened window (Figure 3.5), chose among the distance calculation options (currently only the first options are supported) that are given in the right side of the window and set a maximum number of represented service instances (affects the level of nesting in the non-functional abstractions hierarchies). Then, press the `Produce Hierarchy` button. The produced hierarchies are shown in two window panels. The produced hierarchies can be stored in the Service Base by clicking on the `Store Hierarchies` button.

Figure 3.6 shows the `Query Engine` tab. You can provide to the `Query Engine` an input file that contains an SBQL query by clicking on the `Select File` button. The query is shown on the right side

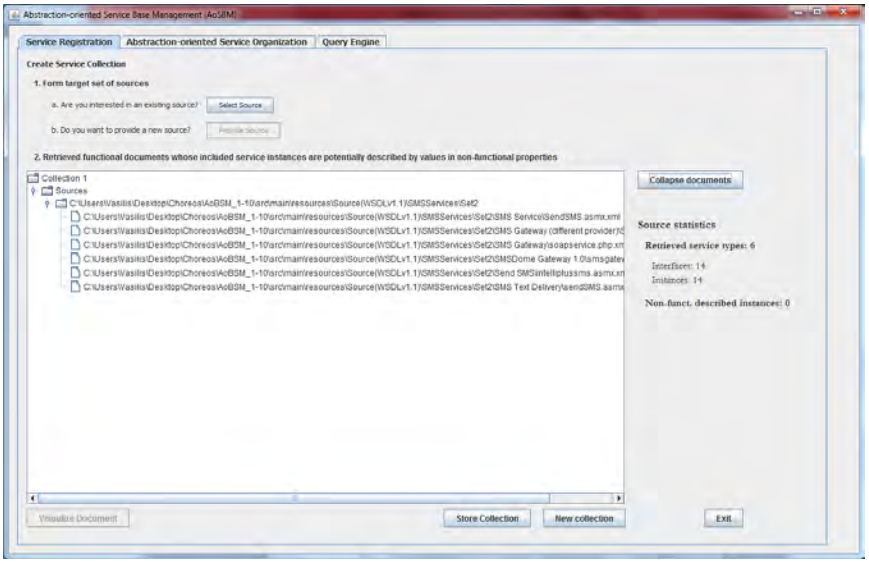

Figure 3.3: Using the AoSBM `Service Registration` tab.

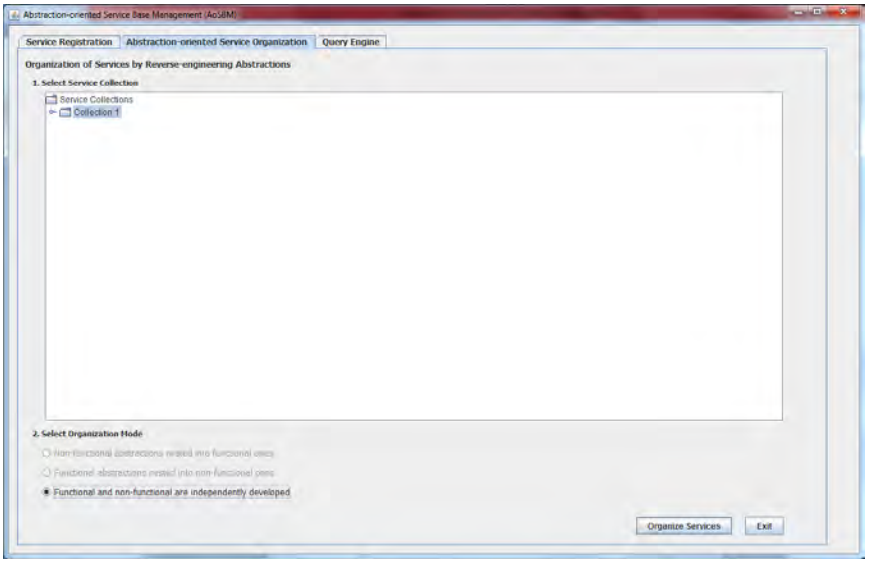

Figure 3.4: Using the AoSBM `Abstraction-oriented Service Organization` tab (1).

of the tab. Then, by clicking on the `Execute Query` button the `Query Engine` parses the input file and creates the SQL statement to by executed, executes it, reconstructs and visualizes the abstractions that meet the query specification.

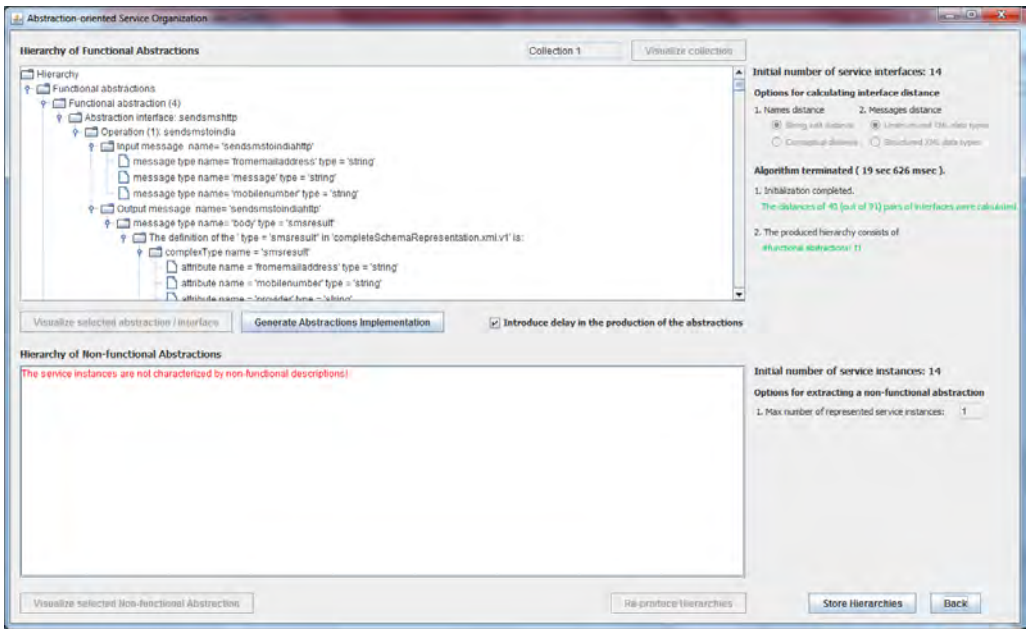

Figure 3.5: Using the AoSBM `Abstraction-oriented Service Organization` tab (2).

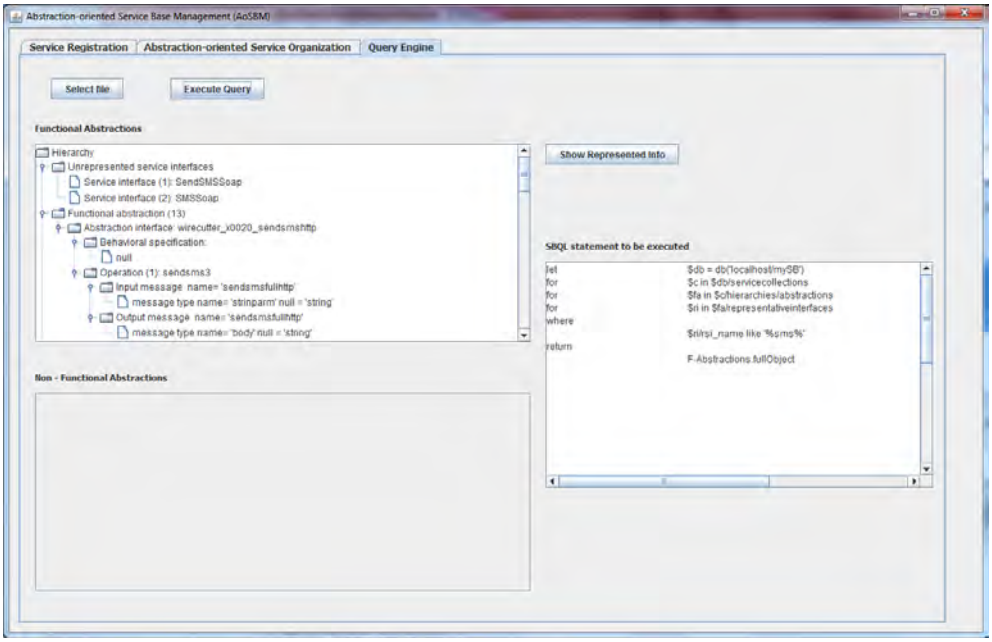

Figure 3.6: Using the AoSBM `Query Engine` tab.

3.3.2. Discovery Plugin User Manual

The Discovery Plugin Manager provides unified access to service discovery protocols present in the CHOReOS environment and essentially acts as an intermediate layer between them and the Abstraction-oriented Service Base. Via the Plugin Manager, the AoSBM gets populated with services discovered by these heterogeneous service discovery protocols. The Plugin Manager interfaces with these protocols via plugins.

Install Guidelines

```
Plugin Manager - INSTALLATION GUIDE - 11/04/12

Written by Kostas Lekkas, Thanassis Parathyras (VTRIP)

lekkas@vtrip.net
aparathyras@vtrip.net

== WHAT IS IT? ==

The Plugin Manager is a component of the extensible service discovery that enables
incorporating support for any current or future service discovery solution applying
to IoBS and IoTS. It has been described in detail at D3.1 and D3.2.1. The 'AoSBM'
and 'plugin_manager' subpackages contain the sources for the Discovery Plugin
Framework, while the 'DummyActiveRegistry' and 'DummyPlugin' subpackages contain the
sources for a test case of the aforementioned framework. For more information on
Plugin Development please refer to D3.1, D3.2.1 and the Javadoc of the sources.

== REQUIREMENTS ==

The project developed with Eclipse 3.6.2, Maven 3 and JDK 6. Please make sure that
you use compatible software versions.

== CONFIGURING IT ==

For the test use case, sample configuration files can be found at the project
directories. See the 'RUNNING IT' section below.

== COMPILING IT ==

Enter the 'trunk/extensible-service-discovery/plugin_manager' folder and enter the following
    command:

mvn clean compile assembly:single

== RUNNING IT ==

In order to run test use case using Dummy Registry & Plugin, enter the 'PluginManager'
folder and enter the following command:

java -cp target/PluginManager-0.0.1-SNAPSHOT-jar-with-dependencies.jar Main plugin_conf/plugin.
    properties dummy_registry_services/

== VERIFYING IF IT WORKS ==

If everything works OK, the console output will look like this:

$ java -cp target/PluginManager-0.0.1-SNAPSHOT-jar-with-dependencies.jar Main plugin_conf/plugin.
    properties dummy_registry_services/
14:13:41.277 [pool-1-thread-1] INFO o.o.c.e.D.ConnectionAcceptThread - new thread
14:13:41.277 [main] INFO Main -
--Test 1: load plugins from file--
14:13:41.277 [main] INFO o.o.c.e.plugin_manager.PluginManager - Properties file is loaded.
14:13:41.277 [main] INFO o.o.c.e.plugin_manager.PluginManager - Added new class type:org.ow2.
    choreos.executable_service_discovery.DummyPlugin.DummyPlugin
14:13:41.277 [main] INFO Main -
--Test 2: explicitly load plugin--
14:13:41.277 [main] INFO o.o.c.e.plugin_manager.PluginManager - Plugin org.ow2.choreos.
    executable_service_discovery.DPWSPlugin not found
14:13:41.277 [main] INFO Main -
```

```
--Test 3: see available plugins--
14:13:41.277 [main] INFO Main - Available plugin: org.ow2.choreos.executable_service_discovery.
    DummyPlugin.DummyPlugin
14:13:41.277 [main] INFO Main -
--Test 4: see loaded plugins--
14:13:41.277 [main] INFO Main - Loaded plugin: org.ow2.choreos.executable_service_discovery.
    DummyPlugin.DummyPlugin
14:13:41.277 [main] INFO Main -
--Test 7: create plugin instance--
14:13:41.277 [main] INFO o.o.c.e.DummyPlugin.DummyPlugin - plugin localhost:31337 initialized;
14:13:41.277 [main] INFO o.o.c.e.DummyPlugin.DummyPlugin - start called
14:13:41.292 [main] INFO o.o.c.e.plugin_manager.PluginManager - Plugin instance
    localhost:31337started
14:13:41.308 [main] INFO Main -
--Test 8: get list of plugin instances for plugintype--
14:13:41.308 [main] INFO Main - Instantiated plugin: localhost:31337
14:13:41.308 [pool-4-thread-1] INFO o.o.c.e.DummyPlugin.RegistryListener - Inside run() of
    RegistryListener
14:13:41.308 [Thread-1] INFO o.o.c.e.plugin_manager.PluginManager - Started reader thread
14:13:41.308 [pool-3-thread-1] INFO o.o.c.e.D.ConnectionAcceptThread - new entry: C:\Documents and
     Settings\Lekkas\Eclipse_workspace\PluginManager\dummy_registry_services\service1.wsdl
28b9a1ee52e9e1677d4064dbdc3ec414
14:13:41.355 [Thread-0] INFO o.o.c.e.D.ConnectionAcceptThread - New connection: /127.0.0.1:1622
14:13:41.371 [pool-3-thread-1] INFO o.o.c.e.D.ConnectionAcceptThread - new entry: C:\Documents and
     Settings\Lekkas\Eclipse_workspace\PluginManager\dummy_registry_services\service2.wsdl
5297ca6fe8ce746f00e5e405723baf4e
14:13:43.371 [pool-3-thread-1] INFO o.o.c.e.D.ConnectionAcceptThread - Sent msg
14:13:43.371 [pool-3-thread-1] INFO o.o.c.e.D.ConnectionAcceptThread - Sent msg
Root element registrymsg
NEWENTRY
14:13:43.402 [Thread-1] INFO MockAoSBM - register() called: registry: localhost:31337
Root element registrymsg
NEWENTRY
14:13:43.449 [Thread-1] INFO MockAoSBM - register() called: registry: localhost:31337
14:13:51.308 [main] INFO Main -
--Test 9: destroy plugin instance--
14:13:51.308 [main] INFO o.o.c.e.DummyPlugin.DummyPlugin - stop called
14:13:51.308 [Thread-1] INFO o.o.c.e.plugin_manager.PluginManager - Reader Thread interrupted:
14:13:51.308 [pool-4-thread-1] ERROR o.o.c.e.DummyPlugin.RegistryListener - socket closed
```

Listing 3.10: Plugin Manager User Manual

3.3.3. Things Discovery Plugin User Manual

The Things Discovery component provides two main functionalities: registering things-based services, i.e., sensing and actuating services provided by smart devices, and finding the most appropriate of those services to answer a things-based query. There are three components involved: the Things Registration Manager which determines whether or not the things-based services should be registered; the Registry Manager which stores registered services metadata; and the Things Query Manager which handles things-based queries.

Install Guidelines

```
Things Registration Manager - INSTALLATION GUIDE - 11/04/2012
Written by Hachem Sara inria
mail: sara.hachem@inria.fr

==WHAT IS IT?==
The Things Registration Manager component should be accessed by services on mobile phones to
    determine, in a non-deterministic way
if devices should register their services or not. The component can be downloaded from Maven as a
    jar file. The source code is available at:
"http://forge.ow2.org/projects/choreos/"


==REQUIREMENTS==

#1 Java 6
#2 Apache Maven
#3 Mathematica

The Things Registration Manager uses four libraries:

- jUnit-4.7:
needed for jUnit tests.
can be found on maven repositories.

- jena-2.6.4:
Needed to handle ontologies and RDF files.
Source: http://sourceforge.net/projects/jena/files/Jena/Jena-2.6.4/
Can be found on maven repositories.

- json_simple
Needed to convert HashMaps and ArrayLists to JSON object in order to pass them as parameter to the
     RESTful services.
Source: http://code.google.com/p/json-simple/
Can be found on maven repositories.

- IoTCommonObjects V1.4
Needed for service access, sensor data description, common classes. Deployed on: "http://maven.
    petalslink.com".
Source: committed the project to the OW2 repository in Extensible-Service-Discovery.

- integralLists2905.m file
the file is used by Mathematica to compute some needed values. The file has has been added to the
    top directory and
the path in the code points to its location there.
Path: ./integralLists2905.m

==CONFIGURING IT==

The component uses a RESTful service, provided in choreos/trunk/extensible-service-discovery/
    registry_manager.
 and it assumes that the service has been deployed on the the following address:
 "http://things.vtrip.net:8080/RegistryManager-2.0/rest/"

It also uses Mathematica software to compute some values, the path of the software can be passed
    as an argument if a default
path is not specified in the code. the property to set is "org.ow2.choreos.mathematicaAppDirPath".
e.g. -Dorg.ow2.choreos.mathematicaAppDirPath="/Applications/Mathematica.app/Contents/MacOS/"
If no path is provided the process will stop.
```

```
The RDF and Mathematica script files should be placed in the top directory or their paths should
be specified.
The property to set for the Mathematica script file is "org.ow2.choreos.scriptDirPath"
The property to set for the RDF file is "org.ow2.choreos.registration.rdfFilePath"

==VERIFYING IF IT WORKS==

Several test classes have been added to /test/java under the src folder in the project.
```

Listing 3.11: Registration User Manual

```
Registry Manager - INSTALLATION GUIDE - 11/04/2012
Written by Hachem Sara - Inria
mail:sara.hachem@inria.fr

==WHAT IS IT?==

The registry manager provides REST servlets that store registration metadata in memory, remove
    registration metadata from memory,
and provides stored registration metadata. The component can be downloaded from Maven as a war
    file. The source code is available at:
"http://forge.ow2.org/projects/choreos/"

==REQUIREMENTS==

#1 Java 6
#2 Apache Maven
#3 APACHE tomcat

The registry Manager component uses three libraries:

- json_simple
needed to convert HashMaps and ArrayLists to JSON object in order to pass them as parameter to the
     RESTful services.
source: http://code.google.com/p/json-simple/
can be found on maven repositories

- javaee-api-6.0.jar
source: http://download.java.net/maven/2/javax/javaee-api/6.0/
can be found on maven

- IoTCommonObjects V1.4
Needed for service access, sensor data description, common classes. Deployed on: "http://maven.
    petalslink.com".
Source: committed the project to the OW2 repository in Extensible-Service-Discovery


==CONFIGURING IT==

The web service should be deployed locally or on a server with Apache tomcat.
The service is deployed on the following address:
 "http://things.vtrip.net:8080/RegistryManager-2.0/rest/"

==VERIFYING IF IT WORKS==

Several test classes have been added to /test/java under the src folder in the project.
```

Listing 3.12: Registry Manager User Manual

```
Things Query Manager - INSTALLATION GUIDE- 11/04/2012
Written by Hachem Sara - Inria
mail:sara.hachem@inria.fr

==WHAT IS IT?==

The things Query Manager is responsible for receiving, expanding, treating queries for real world
    data measurements
forwarded by user applications. The component can be downloaded from Maven as a jar file. The
    source code is available at:
"http://forge.ow2.org/projects/choreos/"


==REQUIREMENTS==

#1 Java 6
#2 Apache Maven
#3 Mathematica

The Things Query Manager component utilizes four libraries:

- json_simple
Needed to convert HashMaps and ArrayLists to JSON object in order to pass them as parameter to the
     RESTful services.
Source: http://code.google.com/p/json-simple/
Can be found on maven repositories

- IoTCommonObjects V1.4
Needed for service access, sensor data description, common classes. Deployed on: "http://maven.
    petalslink.com".
Source: committed the project to the OW2 repository in Extensible-Service-Discovery

- Registration V 2.0
Deployed on: "http://maven.petalslink.com".
Source : committed the project to the OW2 repository in Extensible-Service-Discovery

- Composition Manager V1.2
Deployed on Petalslink maven repository
Source : committed the project to the OW2 repository in Executable-Service-Composition,

- sensorsAndActuators.rdf and unitCoversion.rdf
The component also uses two RDF files that should be placed at the root, and the path in the code
    points to its location at the root.
./sensorsAndActuators.rdf and ./unitConversion.rdf

==CONFIGURING IT==

The component uses a RESTful service, provided in choreos/trunk/extensible-service-discovery/
    registry_manager.
It assumes that the service has been deployed on the following address:
 "http://things.vtrip.net:8080/RegistryManager-2.0/rest/"

The RDF files should be placed in the top directory

==VERIFYING IF IT WORKS==

A demo class (org.ow2.choreos.querydemo.TestQueryMain) has also been added to show how a query is
    passed to the middleware.
```

Listing 3.13: Things Query Manager User Manual

3.4. Cloud & Grid

3.4.1. Node Pool Manager and Service Deployer User Manual

Node Pool Manager enables you to create virtual machines as nodes in a cloud infra-structure and retrieve nodes with specific configuration requirements by using a REST API. At the moment, Amazon Web Services (AWS EC2) and OpenStack are supported as cloud infra-structure.

Service Deployer enables you to easily deploy web services into cloud nodes. At the moment there are two deploy options supported: TOMCAT and COMMAN_DLINE services. TOMCAT services are WAR files intended to run in Tomcat. COMMAN_DLINE services are executed by runnable jar files that must contain the services themselves and the main method to start the services. Current work is on progress to enable the deployment of coordination delegates on EasyESB nodes.

Since Node Pool Manager and Service Deployer are intended to run in the same node and have similar requirements, we have unified the installation guide of these components.

Install Guidelines

```
NODE POOL MANAGER / SERVICE DEPLOYER - INSTALLATION GUIDE
October, 02. 2012
Written by Leonardo Leite (USP)
leonardofl87@gmail.com
== REQUIREMENTS ==

Before you run Node Pool Manager and Service Deployer, you will need:

#1 Java 6 (we are using OpenJDK)

#2 Maven 3 (http://maven.apache.org/download.html)

#3 An AWS account or an OpenStack identification

Node Pool Manager will use the AWS EC2 service or OpenStack infrastructure, which controls the
    management of virtual machines. You can create an AWS account at http://aws.amazon.com.
    Whereas using AWS EC2 is a paid services, using OpenStack requires you to deploy all the
    OpenStack infrastructure in your organization.

#4 An Chef account

Node Pool Manager and Service Deployer will properly configure the nodes using Chef, an open-
    source configuration management system. With Chef you can specify a resources set to be
    deployed into cloud nodes. These resources are described by a Ruby-like DSL (Domain Specific
    Language), and include: systems, files, scripts execution, and others.

You can setup your own Chef server, or create an Hosted Chef server account. Hosted Chef is
    offered by Opscode (http://www.opscode.com/). Although Hosted Chef frees you from setting the
    Chef Server in the infrastructure of your organization, it allows only a limited number of
    nodes to be managed by Chef.

== CONFIGURING IT ==

Open the folder ServiceDeployer/src/main/resources, and create a servicedeployer.properties file
    by copying the servicedeployer.properties.template file. The new properties file must be
    created in the same folder.

Open the just created properties file and edit it as follows:

NODE_POOL_MANAGER_PORT: the TCP port that will serve the Node Pool Manager
SERVICE_DEPLOYER_PORT: the TCP port that will serve the Service Deployer
NODE_SELECTOR:
CLOUD_PROVIDER:
FIXED_VM_IP: the IP of the virtual machine to test purposes; not necessary
FIXED_VM_HOSTNAME: the host name of the virtual machine to test purposes
FIXED_VM_PRIVATE_SSH_KEY: the location of the file with the private key to access the above
    virtual machine by SSH
FIXED_VM_USER: the user to log in the virtual machine to test purposes
AMAZON_ACCESS_KEY_ID: credentials of your AWS account
```

```
AMAZON_SECRET_KEY: credentials of your AWS account
AMAZON_KEY_PAIR: the name of the key pair generated on the AWS web console to access the nodes by
    SSH
AMAZON_PRIVATE_SSH_KEY: the location of the file with the private key corresponding to the above
    key pair
CHEF_CONFIG_FILE: your Chef configuration file location; usually the .chef/knife.rb within your
    chef repo.
CHEF_REPO: your Chef repo folder, where you store Chef recipes.

Example:
NODE_POOL_MANAGER_PORT=9100
SERVICE_DEPLOYER_PORT=9101
NODE_SELECTOR=ROUND_ROBIN
CLOUD_PROVIDER=AWS
FIXED_VM_IP=192.168.56.102
FIXED_VM_HOSTNAME=choreos-node
FIXED_VM_PRIVATE_SSH_KEY=/home/leonardo/.ssh/nopass
FIXED_VM_USER=choreos
AMAZON_ACCESS_KEY_ID=AKIAIIT213ISasdSECRETEFJH6Q
AMAZON_SECRET_KEY=N+KzHQITasdIS123ALSOwzAj9MiSECRETE0UPuwyD
AMAZON_KEY_PAIR=leofl
AMAZON_PRIVATE_SSH_KEY=/home/leonardo/.ssh/leofl.pem
CHEF_CONFIG_FILE=/home/leonardo/chef/chef-repo/.chef/knife.rb
CHEF_REPO=/home/leonardo/chef/chef-repo

Make sure your knife.rb file be something like:

log_level :info
log_location STDOUT
node_name "lleite"
client_key "#{current_dir}/lleite.pem"
validation_client_name "choreos-verao-validator"
validation_key "#{current_dir}/choreos-verao-validator.pem"
chef_server_url "https://api.opscode.com/organizations/choreos-verao"
cache_type 'BasicFile'
cache_options( :path => "#{ENV['HOME']}/.chef/checksums")
cookbook_path ["#{current_dir}/../cookbooks"]

Here, "lleite" is also my Opscode user name, and "choreos-verao" is the organization name
    configured on Hosted Chef. And do not forget the "cookbook_path" property.

You will also need to upload to your Chef Server all the cookbooks from our cookbook folder:
    https://github.com/choreos/choreos_middleware/tree/master/chef-repo/cookbooks.

== COMPILING IT ==

After installing Maven 3, open the terminal at the choreos_middleware folder, and run the build.sh
     script.

It can take several minutes.

== RUNNING IT ==

After compiling the project, to run both Node Pool Manager and Service Deployer you have just to
    run the main method on the org.ow2.choreos.servicedeployer.rest.Servers class.

You must run the system using Java 6.

This task can be easier accomplished if you import the ServiceDeployer project in the Eclipse IDE.

After importing the project, open the menu Window>>Preferences>>Java>>Build Path>>Classpath
    variables, and set the M2_REPO variable pointing to your Maven repository folder, usually the
    .m2/repository folder within your home folder.

Obs: we have used the OpenJDK JVM and the Eclipse Indigo version.

If you successfully start the Node Pool Manager and the Service Deployer, you must see the
    following messages on the console:
Node Pool Manager has started [http://localhost:9100/nodepoolmanager/]
Service Deployer has started [http://localhost:9101/servicedeployer/]

== VERIFYING IF NODE POOL MANAGER WORKS ==
```

```
First go to your cookbook folder within your chef repo. Copy the "getting-started" folder and past
     it, creating a second folder called "getting-started2". Open the file recipes/default.rb and
    change the line "template "#{ENV['HOME']}/chef-getting-started.txt" do" by "template "#{ENV['
    HOME']}/chef-getting-started2.tx" do".

Upload this new cookbook to your chef server.

Start Node Pool Manager as earlier described.

Open Firefox and install the Poster plugin, which enables you to send HTTP requisitions.

With Poster, send the following requisition:

HTTP method: POST
URI: http://localhost:9100/nodepoolmanager/nodes/configs
Content-Type: application/xml
Body: <config><name>getting-started2</name></config>

Change the port number if necessary.

The response must take some time to be delivered. Take note the IP property of the node XML
    representation sent in the response body. You must use this IP to login with SSH in this VM,
    and verify if the home folder contains the getting-started2.txt file. If so, it worked!

Obs: if you have configure NODE_SELECTOR=ROUND_ROBIN, you must be sure there is some node already
    created on your cloud environment before post the request.

== VERIFYING IF SERVICE DEPLOYER WORKS ==

With Poster, the Firefox plugin, send the following request:

HTTP method: POST
URI: http://localhost:9101/servicedeployer/services
Content-Type: application/xml
Body:
<serviceSpec>
  <name>my_war</name>
  <type>TOMCAT</type>
  <codeUri>LINK</codeUri>
</serviceSpec>

Change the port numbers if necessary. Replace "LINK" by an URL pointing to a WAR file. To learn
    how to build a WAR file containing a web service: http://ccsl.ime.usp.br/redmine/projects/
    bailevv/wiki/Jaxws_tomcat.

The application provided by the WAR file must be accessible by the address http://host:8080/
    WAR_FILE_NAME.
Such URL will be informed in the HTTP response body, and the host will be chosen by the Node Pool
    Manager.

If you prefer to deploy a JAR, requests the following:

HTTP method: POST
URI: http://localhost:9101/servicedeployer/services
Content-Type: application/xml
Body:
<serviceSpec>
  <name>my_jar</name>
  <type>JAR</type>
  <codeUri>LINK</codeUri>
  <endpointName>endpointName</endpointName>
  <port>portNumber</port>
</serviceSpec>

The "endpointName" and "port" elements must be complaint with what the JAR code does.
So, the service must be available at http://host:portNumber/endpointName.
```

Listing 3.14: Node Pool Manager and Service Deployer User Manual

3.4.2. Enactment Engine User Manual

The Enactment Engine implements a RESTful API that enables the enactment of choreographies. The client sends a choreography specification to the Enactment Engine, and the Enactment Engine will coordinate the necessary calls to Service Deployers and to Node Pool Managers to perform the choreography deployment. The client receives a response with information about the deployed services, such as their addresses. Within the CHOReOS development process, the Synthesis Process is the Enactment Engine client. Current implementation still enables communication only with a single pair of Service Deployer and Node Pool Manager instances. Only SOAP-based services are acceptable in the moment to be deployed as a choreography.

Install Guidelines

```
ENACTMENT ENGINE - INSTALLATION GUIDE
October, 02. 2012
Written by Leonardo Leite (USP)
leonardofl87@gmail.com
== REQUIREMENTS ==

Before you run Node Pool Manager and Service Deployer, you will need:

Service Deployer and Node Pool Manager properly running.

== CONFIGURING IT ==

Open the folder EnactmentEngine/src/main/resources, and create
a enactmentengine.properties file by copying the
enactmentengine.properties.template file. T
he new properties file must be created in the same folder.

Open the just created properties file and edit it as follows:

ENACTMENT_ENGINE_PORT=9102
NODE_POOL_MANAGER_URI=http://localhost:9100/nodepoolmanager
SERVICE_DEPLOYER_URI=http://localhost:9101/servicedeployer

You may change the values according to your Node Pool Manager
and Service Deployer configurations.

== COMPILING IT ==

After installing Maven 3, open the terminal at the choreos_middleware folder,
and run the build.sh script.
It can take several minutes.

== RUNNING IT ==

After compiling the project, to run the Enactment Engine you have
just to run the main method on the
org.ow2.choreos.enactmentengine.rest.EnactEngServer class.

You must run the system using Java 6.
This task can be easier accomplished if you import
the EnactmentEngine project in the Eclipse IDE.

After importing the project, open the menu
Window>>Preferences>>Java>>Build Path>>Classpath
variables, and set the M2_REPO variable pointing to your Maven
repository folder, usually the .m2/repository
folder within your home folder.

Obs: we have used the OpenJDK JVM and the Eclipse Indigo version.

If you successfully start the Node Pool Manager and the Service Deployer,
you must see the following messages on the console:
Enactment Engine has started [http://localhost:9102/enactmentengine/]
```

Listing 3.15: Enactment Engine User Manual

3.4.3. Storage Factory User Manual

The Storage Factory implements a RESTful API offering storage provisioning services to other CHOReOS Middleware components. Clients should inform the desired storage type, e.g. “MySQL”, and an identifier (UUID). This identifier is the key to that particular storage and can be shared among services that need to share the same storage area. The Storage Factory asks the Node Pool Manager for a node with the needed configuration, and the client receives a CHOReOS Node with the desired storage system deployed on it and the credentials to access the storage system. At the moment, just MySQL is supported as storage system.

Install Guidelines

```
STORAGE FACTORY - INSTALLATION GUIDE
26/03/12
Written by Leonardo Leite (USP)
leonardofl87@gmail.com
== REQUIREMETNS ==

Before you run Storage Factory, you will need Node Pool Manager running.
For convenience, we use the Eclipse-IDE but this is not a strong requirement as our code is IDE
    agnostic.

== CONFIGURING IT ==

As in Node Pool Manager, create the storagefactory.properties file based on the storagefactory.
    properties.template file.

Fill it as the example, adapting when necessary:

STORAGE_FACTORY_PORT=9102
NODE_POOL_MANAGER=http://localhost:9100/nodepoolmanager
CHEF_SERVER_URL=http://aguia1.ime.usp.br:4000
CHEF_CONFIG_FILE=/home/leonardo/chef/chef-repo/.chef/knife.rb
CHEF_USER=leofl
CHEF_USER_KEY_FILE=/home/leonardo/chef/chef-repo/.chef/leofl.pem
CHEF_REPO=/home/leonardo/chef/chef-repo

You will need also to upload the "mysql" cookbook to your chef repo. To download the cookbook in
    your workstation: "knife cookbook site install mysql".


== COMPILING IT ==

Just open the terminal at the project folder, and type "mvn eclipse:eclipse".

You need maven 3 in this step. It can take several minutes.

Before run Storage Factory, you have also to add the NodePoolManager project as a StorageFactory
    dependency.
You can do this at the "Projects" tab in the Build Path project configuration.

== RUNNING IT ==

After compiling the project, you have just to run the main method on the eu.choreos.storagefactory
    .rest.StorageFactoryStandaloneServer class.

You can proceed in the same way as indicated in the Node Pool Manager installation guide.

If you successfully start Storage Factory, you must see the "Starting Storage Factory..." message
    on the console.


== VERIFYING IF IT WORKS ==

With Poster, the firefox plugin, send the following requisition:

HTTP method: POST
URI: http://localhost:9102/storagefactory/storages
Content-Type: application/xml
```

```
Body: <storageNodeSpec><uuid>choreos</uuid><type>MySQL</type></storageNodeSpec>

Change the port numbers if necessary.

After receive the reply, check which was the new VM created on AWS console. This step can take
    several minutes.

You must login with SSH in this newly created VM, and type the following:

\$mysql -u choreos -h localhost -p

Enter the password (choreos), and if the mysql prompt appears it is working!

IMPORTANT NOTE: Currently, the user is not able to remotely connect to a data base created in a
    AWS node, although the remote connection is possible if Node Pool Manager uses the "fixed
    cloud provider", which uses a virtual machine instance running in the development machine.
    Anyway, it is an issue we must still work on.
```

Listing 3.16: Storage Factory User Manual

CHOReOS Governance and V&V Framework User Manuals

The reference implementation of the CHOReOS Governance and V&V Framework is available in [9].

3.5. CHOReOS Development and Runtime Governance

3.5.1. Rehearsal

Rehearsal is a framework for design-time testing of choreography, in particular adapting a Test-Driven Development (TDD) approach to the scenario of ULS choreography development.

Install Guidelines

```
REHEARSAL INSTALLATION GUIDE - 10/03/2012
Written by Paulo Moura (USP)
pbmoura@ime.usp.br
== WHAT IS IT? ==
Rehearsal is an integrated framework supporting the automated testing
of choreographies.
Currently, Rehearsal is spitted in two different projects:
one including the features for correctness tests, supporting TDD;
and the other concerning with scalability tests.
== REQUIREMENTS ==
 1. java jdk 1.6
 2. maven 3.0.4
== COMPILING IT ==
Both the projects are available at:
svn://svn.forge.objectweb.org/svnroot/choreos/trunk/governance/tdd
in directories 'rehearsal' and 'scalability_explorer'.
They can be compiled with 'mvn compile'
== RUNNING IT ==
As Rehearsal is a framework, it is not properly run, but is supposed
to be used by other projects.
The simplest way to do this is by installing Rehearsal in your local
maven repository and including it as a dependence in your project.
The installation can be done with 'mvn install'
To establish the dependencies, you should add the following to your pom.xml.
for correctness tests:
   <dependency>
     <groupId>eu.choreos</groupId>
     <artifactId>rehearsal</artifactId>
     <version>0.13</version>
     <scope>jar</scope>
   </dependency>
for scalability tests:
   <dependency>
     <groupId>eu.choreos.vv</groupId>
     <artifactId>scalability_explorer</artifactId>
     <version>1.0-SNAPSHOT</version>
     <scope>jar</scope>
   </dependency>
== VERIFYING IF IT WORKS ==
Check if you can import and use Rehearsal's components in your project.
```

Listing 3.17: Rehearsal User Manual

3.5.2. ServicePot

ServicePot is a UDDI compliant SOA Registry, enhanced with additional information to store Web Service Choreographies. Among the other feature it offers, ServicePot can enable runtime Verification and Validation of Web Services and Choreographies.

Install Guidelines

```
ServicePot
03/07/12
Written by Midhat ALI (Unicam)
midhat.ali@unicam.it
== WHAT IS IT? ==
ServicePot is a UDDI Registry for Web Service Choreographies. It provides functions to publish and
     retrieve Choreographies, in addition to the standard UDDI APIs for Web Services

== REQUIREMETNS ==

Before you compile the project you need to install

#1 Java 1.6

#2 Apache Maven

== CONFIGURING IT ==

ServicePot uses a configuration file located at choreos/trunk/ServicePot/src/main/resources/
    UDDIProxy.properties
This file needs to contain URIs to endpoints of a standard UDDI Registry

== COMPILING IT ==

Compile the project by entering the command "mvn install" at the command prompt. This will
    generate the war file in the servicepot-war/target directory

== RUNNING IT ==

Deploy the war file from dist directory to a tomcat server installation

== VERIFYING IF IT WORKS ==

Service Listing should be available at <ApplicationURL>/servicepot/services
```

Listing 3.18: ServicePot User Manual

3.5.3. Partes

Partes is a framework that tests the participants involved in a choreography in order to discover possible mismatches in their behavior respect to an assigned role. Mismatches can prevent the integration in the choreography bringing to run-time errors.

Install Guidelines

```
Partes - Participant Testing - INSTALLATION GUIDE - 29/06/2012
Written by Francesco DE ANGELIS (Unicam)
francesco.deangelis@unicam.it
== WHAT IS IT? ==

This component provides a test generation strategy for choreography participants. At the moment,
    the strategy starts with a java model of a choreography and brings to a set of SOAPui projects
     for the participants involved.

The implementation of Partes is contained in two maven module:

- partes contains all the steps for test case derivation described in the D4.2.1 deliverable

- partes.ws contains the web services to expose the implementation as a service.


== REQUIREMENTS ==

Before you compile the project you need to install:

#1 Java 6

#3 Java Path Finder (http://babelfish.arc.nasa.gov/trac/jpf) (see later for configuration)

#4 Optional: Tomcat with CXF libraries to deploy the Partes services

#5 Optional: Eclipse (to view the code samples)

#6 Optional: SoapUI (to open the test suites derived by Partes)


== CONFIGURING IT ==

Partes uses Java PathFinder (JPF) as model checker. This requires a machine dependant
    configuration and a "site.properties" file should be placed in the "<user.home>/.jpf/"
    directory of the machine.

Please refer to http://babelfish.arc.nasa.gov/trac/jpf/wiki/install/site-properties for more
    information.

A simple and straightforward site.properties file contains:

jpf-core = ${user.home}/workspaces/workspacePartes01/bpt.jpf-core

jpf-net-iocache = ${user.home}/workspaces/workspacePartes01/bpt.jpf-net-iocache

extensions = ${jpf-core}

to list the directories of the jpf installation. Currently, we use the "core" jpf tools and the "
    jpf-net-iocache" extension.


== COMPILING IT ==

"mvn clean package" compiles the code and produces a jar file for partes and a war file for partes
    .ws

For partes a "fat" jar with all the runtime dependencies is also built.


== RUNNING IT ==
```

```
To run partes form a command line just run the jar that is built the "target" directory

To run partes.ws put the generated war in a Tomcat container to deploy the service

== VERIFYING IF IT WORKS ==

The build process executes some tests that require a full test suite generation including a JPF
    run.
If the maven process build partes and partes.ws without errors the tool is ready to be used.

If the maven process build partes producing errors in the test phase maybe JPF is not configured
    properly in the machine.
To verify the build you can run Partes from the command line providing the necessary inputs.
```

Listing 3.19: Partes User Manual

3.5.4. CRank

CRank implements mechanism for rating both services and service choreography [4]. This information is useful to improve the service selection process and to put in place policies concerning service lifecycle activities related to those services made available by the corresponding service provider. Furthermore such rating can be used in order to deal with properties on the whole choreography (e.g. determining if a choreography is “enactable”).

Install Guidelines

```
CRank - INSTALLATION GUIDE - 17/09/2012
Written by Guglielmo De Angelis (CNR)
guglielmo.deangelis@cnr.isti.it
== REQUIREMENTS ==
These libraries are supposed to be installed in the classpath of the platform
  hosting the "aar" service:
[--DEPENDENCIES--]
<dependencies><dependency>
   <groupId>org.apache.axis2</groupId>
   <artifactId>axis2-kernel</artifactId>
   <version>1.6.2</version></dependency>
 <dependency>
   <groupId>org.apache.axis2</groupId>
   <artifactId>axis2-adb</artifactId>
   <version>1.6.2</version>
 </dependency>
 <dependency>
   <groupId>org.apache.axis2</groupId>
   <artifactId>axis2-adb-codegen</artifactId>
   <version>1.6.2</version></dependency>
 <dependency>
   <groupId>org.apache.axis2</groupId>
   <artifactId>axis2-xmlbeans</artifactId>
   <version>1.6.2</version>
 </dependency>
 <dependency>
   <groupId>rome</groupId>
   <artifactId>rome</artifactId>
   <version>1.0</version>
 </dependency>
 <dependency>
   <groupId>org.jdom</groupId>
   <artifactId>jdom</artifactId>
   <version>1.1</version>
 </dependency>
 <dependency>
   <groupId>la4j</groupId>
   <artifactId>la4j</artifactId>
   <version>0.1.0</version>
 </dependency>
 <dependency>
   <groupId>org.apache.ws.commons.axiom</groupId>
   <artifactId>axiom-impl</artifactId>
   <version>1.2.12</version>
 </dependency></dependencies>

== COMPILING IT ==
 run mvn compile

== CONFIGURING IT ==
To create a service for Axis2 run: mvn package

== RUNNING IT ==
How to deploy the service :
 - Generate the "aar" service from the steps 1-2.
 - Follow the standard procedure for deloying "aar" files into your
   Axis2 distribution. For more information, please visit http://axis.apache.org/axis2/java/core/
```

Listing 3.20: CRank User Manual

3.5.5. Q4BPMN

Q4BPMN (Quality for BPMN) is an approach to directly annotate the BPMN Choreography Diagram with quality requirements the services entering the choreography will have to abide by.

Q4BPMN implements the concepts defined by the Property Meta-Model (PMM) [10] as a UML profile. A detailed description about the approach, and about the usage of the Q4BPMN Profile see [3] [2] [1].

Install Guidelines

For more detailed information, refer to Section 4.1, or to its official website : `http://labse.isti.cnr.it/tools/q4bpmn`.

3.5.6. SLA & Lifecycle Management

The SLA & Lifecycle Management tool is delivered as part of the WP4 prototype. The prototype implementing the SLA & lifecycle manager is called EasierGov. It ensures the management of the lifecycle of resources such services, sla, choreographies, etc. Moreover, it provides an API enabling to synchronize with a service infrastructure (such as EasyESB- XSA) that allows managing services as WSDL and service level agreements as WS-Agreement.

Install Guidelines

```
   CHOReOS Install Files Template

Sla & Lifecycle Management and Governance and V&V Framework - INSTALLATION GUIDE - 08/10/2012

Written by Amira Ben Hamida (Linagora)

nicolas.salatge@linagora.com
julien.lesbegueries@linagora.com
amira.ben-hamida@linagora.com

== WHAT IS IT? ==
The SLA & Lifecycle Management tool is delivered as part of the WP4 prototype. The prototype
    implementing the
SLA & lifecycle manager is called EasierGov. It ensures the management of the lifecycle of
    resources such services,
 SLA, choreographies, etc. Moreover, it provides an API enabling to synchronize with a service
     infrastructure (such as EasyESB- XSA)
 that allows managing services as WSDL and service level agreements as WS-Agreement.

== REQUIREMENTS ==
  1. Download java jdk 1.6
  2. Set JAVA_HOME in your classpath
  3. Copy this endorsed directory in jre library:
  $JAVA_HOME/jre/lib (for linux/windows) and $JAVA_HOME/lib (for mac)
  4. Download maven 2.2.1
  5. Set MAVEN_HOME in your classpath
  6. Set MAVEN_OPTS=-Xms512m -Xmx1024m -XX:PermSize=256m in your classpath

== CONFIGURING IT ==
 1.Download the latest release of EasierGOV from this address:
 http://research.petalslink.org/display/easiergov/Binaries
 2.install it:
  On Windows OS, click on installer
  On Linux OS, java -jar EasierGOV-installer-VERSION.jar

== COMPILING IT ==
Not required. The source code is available here:
http://research.petalslink.org/display/easiergov/Source+Code
Login: anonymous
Password : anonymous

== RUNNING IT ==
Go to the bin directory of project home (shortcut for users of the Windows OS):
 1.On Windows OS, start: ./startup.bat
 2.On Linux OS, start: ./startup.sh
 (remember to grant the write access on this file: chmod a+x startup.sh)

== VERIFYING IF IT WORKS ==
When started successfully the console presented in the figure
(SLA and Lifecycle Management Started Screenshot) appears.
This tool is stopped by hitting the 'X' button of the console.
```

Listing 3.21: SLA and Life Cycle Manager User Manual

Once successfully started, default services are provided: the Admin Service exposed at `http://_host_:_port_/services/adminManager?wsdl`, the Connexion Service exposed at `http://_host_:_port_/services/connexionManager?wsdl`, and the Resources Service exposed at `http://_host_:_port_/services/resourcesManager?wsdl`.

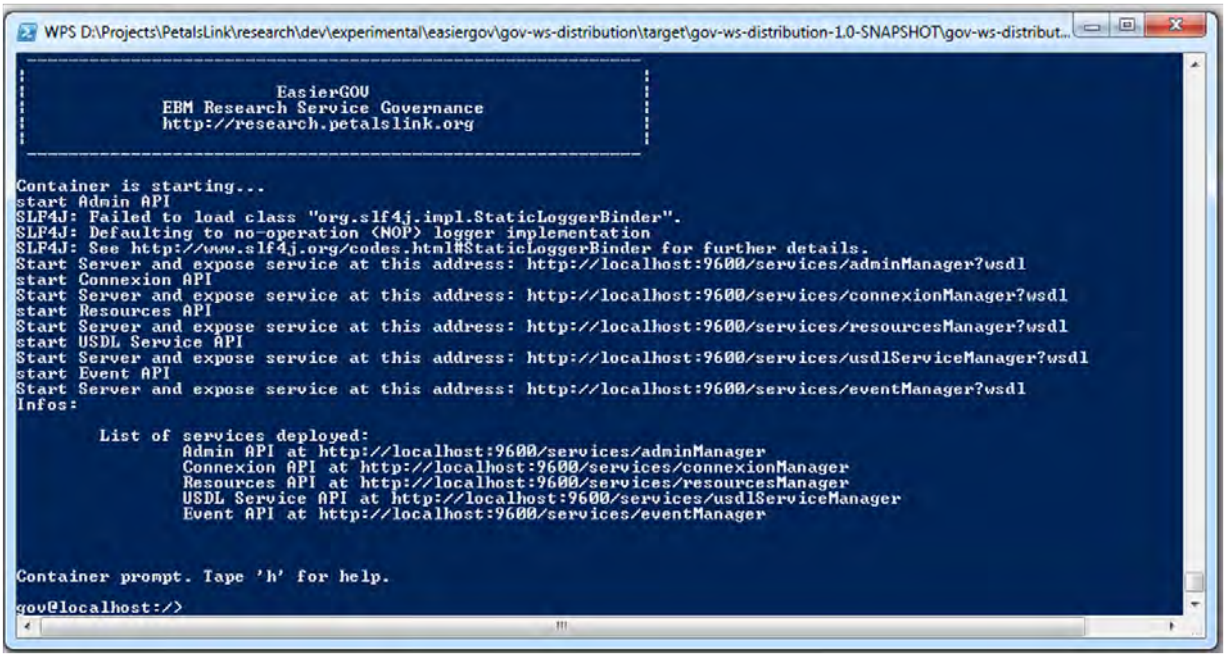

Figure 3.7: SLA & Lifecycle Manager Started Screenshot

The SLA & Lifecycle Manager administration service can be directly [created] or [imported] in topology view of EasiestDEMO Client as shown in the following figure (SLA & Lifecycle Manager Administration Service Screenshot). EasiestDemo is a Web service client developed within Linagora R&D to make more user friendly the administration of the bus `http://research.petalslink.org/display/easiestdemo/Client` (See Section 4.2).

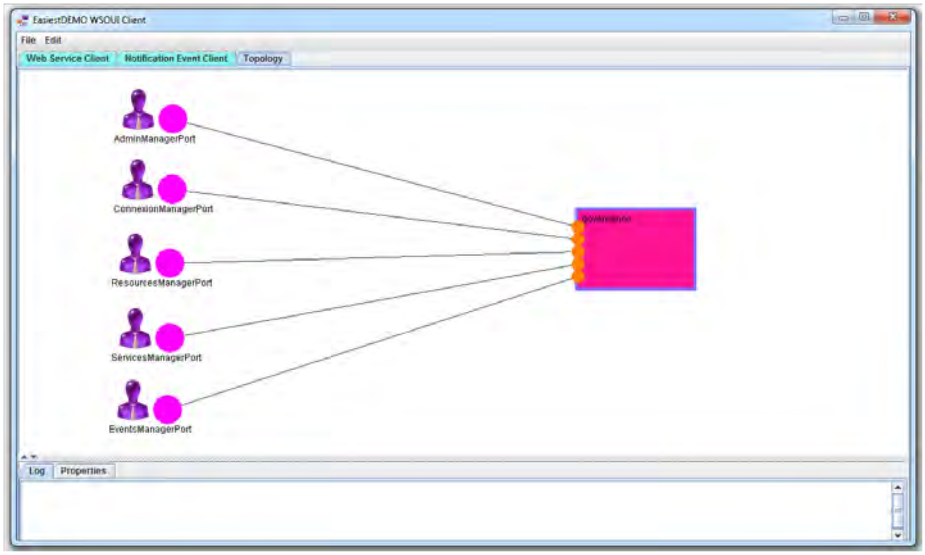

Figure 3.8: SLA & Lifecycle Manager Administration Service Screenshot

3.6. CHOReOS Multi source Monitoring

3.6.1. GLIMPSE

GLIMPSE (Generic fLexIble Monitoring based on a Publish- Subscribe infrastructurE) adopts a model-driven approach, in that, the events coming from Infrastructure Monitoring and Business Monitoring will be correlated in order to infer more complex event pattern happening into the Choreography. A detailed description about the approach, and about the usage of the GLIMPSE can be found in see [9].

Install Guidelines

```
GLIMPSE - INSTALLATION GUIDE - 17/09/2012
Written by Guglielmo De Angelis (CNR)
guglielmo.deangelis@cnr.isti.it

== REQUIREMENTS ==

Before you compile the project you need to install

1# Java 1.6 Runtime
2# A running ActiveMQ instance address.

== CONFIGURING IT ==

In the folder "trunk/monitoring/em/glimpse/dist" you can find the runnable Glimpse package.
The config files is stored into the "configFiles" folder and named as "environmentFile".
The main parameter that you must set is "java.naming.provider.url"
that refers to the activeMQ instance address on which Glimpse will run.

== COMPILING IT ==

Not required, source files can be found into the src package

== RUNNING IT ==

On windows/linux:

java -jar glimpse-manager.jar systemSettings
```

Listing 3.22: Glimpse User Manual

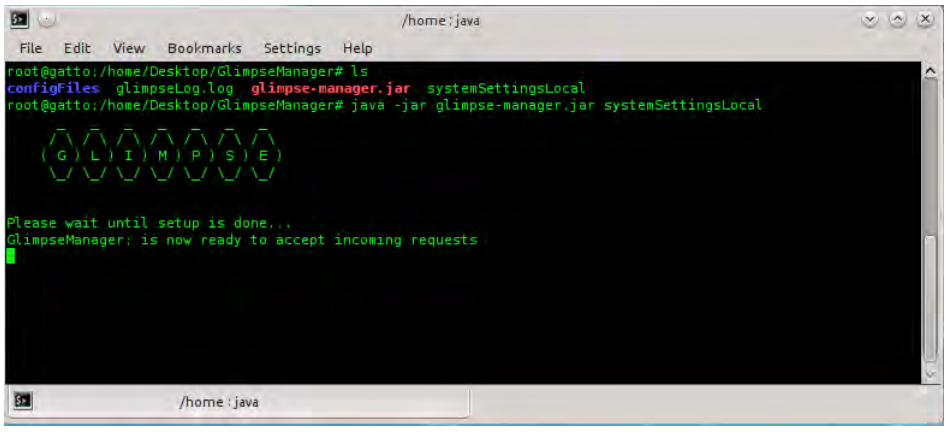

Figure 3.9: Glimpse Started Screenshot

```
== VERIFYING IF IT WORKS ==
```

```
On the console it must appear the message: GlimpseManager: is now ready to accept incoming
    requests
To test it you can also use the glimpse-api package and run one of the test classes (probe/
    consumer)
```

3.6.2. Platform Monitoring

The platform monitor is the component responsible for gathering low-level data, such as disk usage, CPU load etc., and forward this information to the Complex Event Processor component, where it will be further processed to identify eventual problems. This component is comprised of the Ganglia monitoring system (a third-party project) and a daemon (ThresholdEvalDaemon) that collects data from Ganglia and directs it to the Glimpse Complex Event Processor.

Install Guidelines

```
ThresholdEvalDaemon - Platform Monitoring - INSTALLATION GUIDE - 08/10/2012
Written by Nelson Lago (USP)
<lago@ime.usp.br>

== WHAT IS IT? ==
The platform monitor is the component responsible for gathering low-level data,
such as disk usage, CPU load etc., and forward this information to the Complex
Event Processor component, where it will be further processed to identify
eventual problems. This component is comprised of the Ganglia monitoring system
(a third-party project) and a daemon (ThresholdEvalDaemon) that collects data
from Ganglia and directs it to the Glimpse Complex Event Processor.

== REQUIREMENTS ==
Since this component gathers data from Ganglia and directs it to the Glimpse
CEP, Ganglia has to be installed on the local machine and a running instance of
the Glimpse Complex Event Processor must be accessible. As the
ThresholdEvalDaemon is written in Java, a JRE or JDK environment is necessary.
We tested only with OpenJDK6, but it should work with any Java 6 or later
environment.

Since Ganglia is an external project, its code is not bundled here; it should
be installed separately. It should be noted that, in CHOReOS, the Chef
configuration management system is used (within the
EnactmentEngine/NodePoolManager components); accordingly, we do provide a
cookbook for it that automates this installation. The NodePoolManager may be
configured to always deploy ganglia to the nodes it creates if necessary.

There may be issues with older versions of Ganglia; at least version 3.2.0 is
recommended, and we tested only version 3.3.5. The provided chef cookbook makes
use of an unofficial, temporary repository created to support the project.

Maven 3 is required.

== CONFIGURING IT ==
For the ThresholdEvalDaemon to work, you need to define, at least, the
address for the Glimpse CEP on the file src/main/conf/monitoring.properties .
Other parameters may also be tweaked, but the defaults usually suffice. It is
also possible to refine the parameters for what should be considered
"suspicious" by modifying the files in src/main/conf/threshold_specs .
These files currently need to be modified prior to compilation, as they
are bundled into the final packaged jar file.

== COMPILING IT ==
ThresholdEvalDaemon uses Maven 3; after installing maven, "mvn package" should
be enough to create a complete jar with all dependencies.
To compile, please use "mvn -Dmaven.test.skip=true package" .

== RUNNING IT ==
After installing and running Ganglia, the daemon may be run by invoking
"java -jar ChoreosMonitoringService-0.0.1-SNAPSHOT-jar-with-dependencies.jar"
on the command line.

== VERIFYING IF IT WORKS ==
```

```
A simple test to verify the system is working correctly is to start the
daemon (after Ganglia is already running), artificially induce the load
average on the machine to go above one and check whether the Glimpse
CEP receives a corresponding message (this may take up to a couple of
minutes depending on a few parameters).

A simple way to increase load average is this script (CTRL-C stops it):

----
#!/bin/bash

trap "kill -TERM -$$" INT

over() { while true; do :; done }

( over )&
( over )&
( over )&

wait
----
```

Listing 3.23: Platform Monitoring User Manual

3.6.3. Business Service Monitoring

The Business Service Monitoring (BSM) framework is developed within the CHOReOS Monitoring in WP4, it is able to monitor services deployed on a infrastructure of services such as EasyESB. It is also able to manage the agreements negotiated between clients and providers, and communicated from the SLA&Lifecycle Manager (See Section 3.5.6). The BSM is composed of three main components, first, the Data Collector is in charge of collecting all the reports sent by the service infrastructure (XSA). Second, the WSDM Monitoring creates a WSDM (Web Service Distributed Management) non functional endpoint when a functional endpoint is created on the service infrastructure. Third, the SLA Manager is able to send alert when a business exchange in the EasyESB violates the loaded agreement. (We'll call the prototype implementing the BSM, EasierBSM).

Install Guidelines

```
   CHOReOS Install Files Template

Business Service Monitoring and Governance and V&V Framework - INSTALLATION GUIDE - 08/10/2012

Written by Amira Ben Hamida (Linagora)

nicolas.salatge@linagora.com
julien.lesbegueries@linagora.com
amira.ben-hamida@linagora.com

== WHAT IS IT? ==

The Business Service Monitoring is a framework able to monitor services deployed on a
    infrastructure of services such as EasyESB within the XSA.
The implemented prototype of the BSM is called EasierBSM.

== REQUIREMENTS ==
 1. Download java jdk 1.6
 2. Set JAVA_HOME in your classpath
 3. Copy this endorsed directory in jre library:
 $JAVA_HOME/jre/lib (for linux/windows) and $JAVA_HOME/lib (for mac)
 4. Download maven 2.2.1
 5. Set MAVEN_HOME in your classpath
 6. Set MAVEN_OPTS=-Xms512m -Xmx1024m -XX:PermSize=256m in your classpath

== CONFIGURING IT ==
 1.Download the latest release of EasierBSM from this address:
http://research.petalslink.org/display/easierbsm/Binaries
 2.install it:
  On Windows OS, click on installer
  On Linux OS, java -jar EasierBSM-installer-VERSION.jar

== COMPILING IT ==
Not required. The source code is available here:
http://research.petalslink.org/display/easierbsm/Source+Code
Login: anonymous
Password : anonymous

== RUNNING IT ==
Go to the bin directory of project home (shortcut for users of the Windows OS):
 1.On Windows OS, start: ./startup.bat
 2.On Linux OS, start: ./startup.sh
 (remember to grant the write access on this file: chmod a+x startup.sh)

== VERIFYING IF IT WORKS ==
When started successfully the console presented in the figure
(BSM Started Screenshot) appears.
This tool is stopped by hitting the 'X' button of the console (See figure BSM Stopped Screenshot).
```

Listing 3.24: Business Service Monitor User Manual

Once successfully started, EasierBSM exposes by default 2 services: First, the Admin Service described at `http://_host_:_port_/services/bsmadminExternalEndpoint?wsdl` (From EasyESB + BSM specific administration (for SLA, etc.).). Second, the Dispatcher Service described at `http://_host_:_port_`

`/services/DispatcherProviderEndpointClientProxyEndpoint?wsdl`. This service exposes a notification consumer to receive notifications coming from the functional bus.

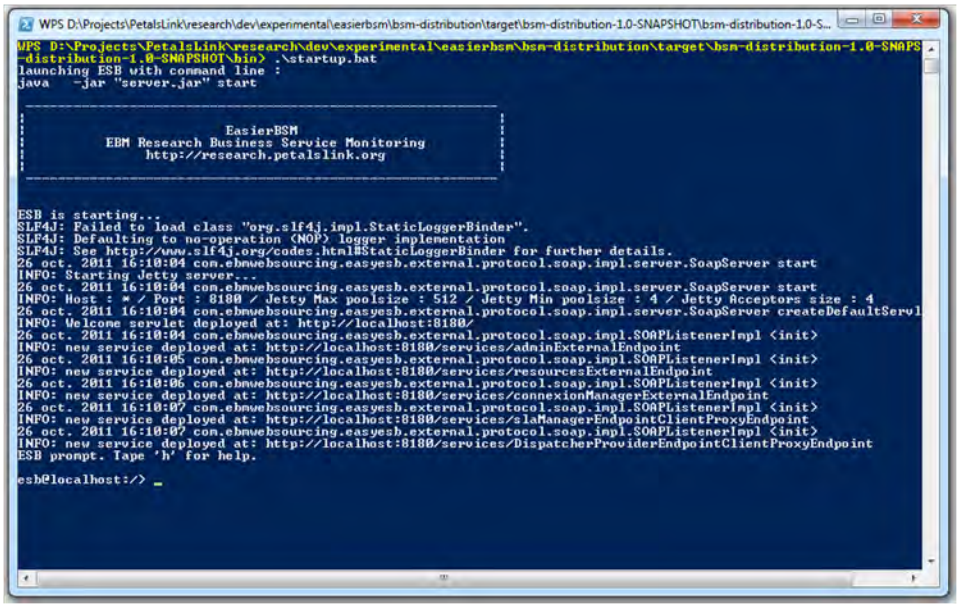

Figure 3.10: BSM Started Screenshot

It can be directly [created] or [imported] in topology view of EasiestDEMO Client as shown in the following figure (BSM Client Screenshot).

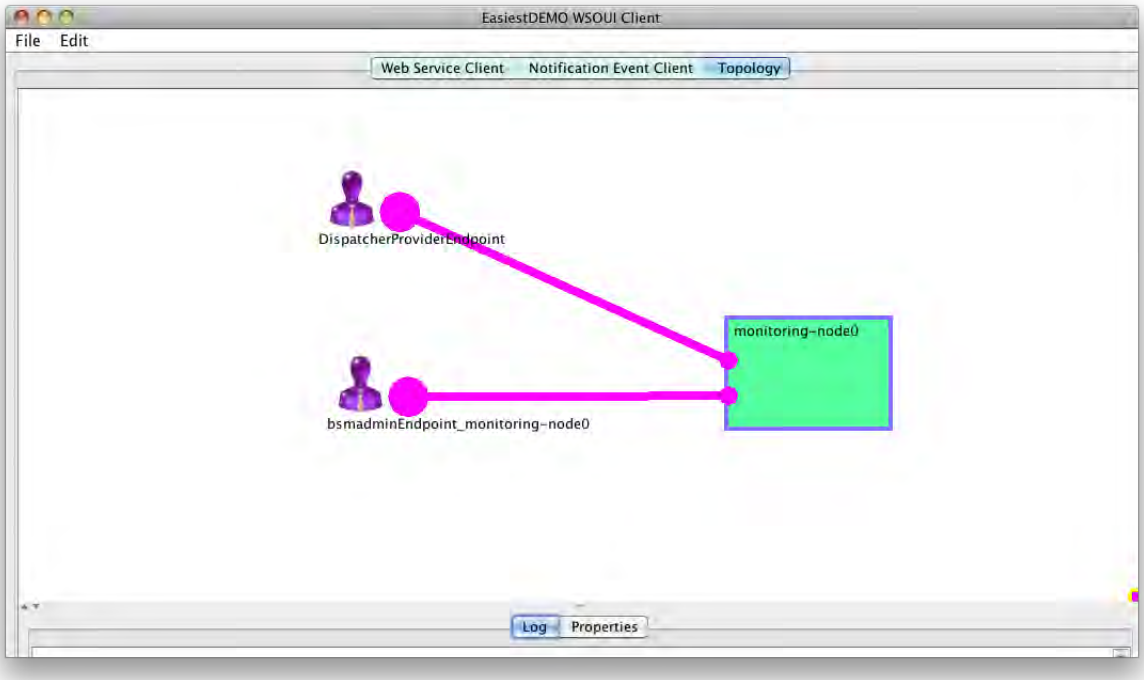

Figure 3.11: BSM Client Screenshot

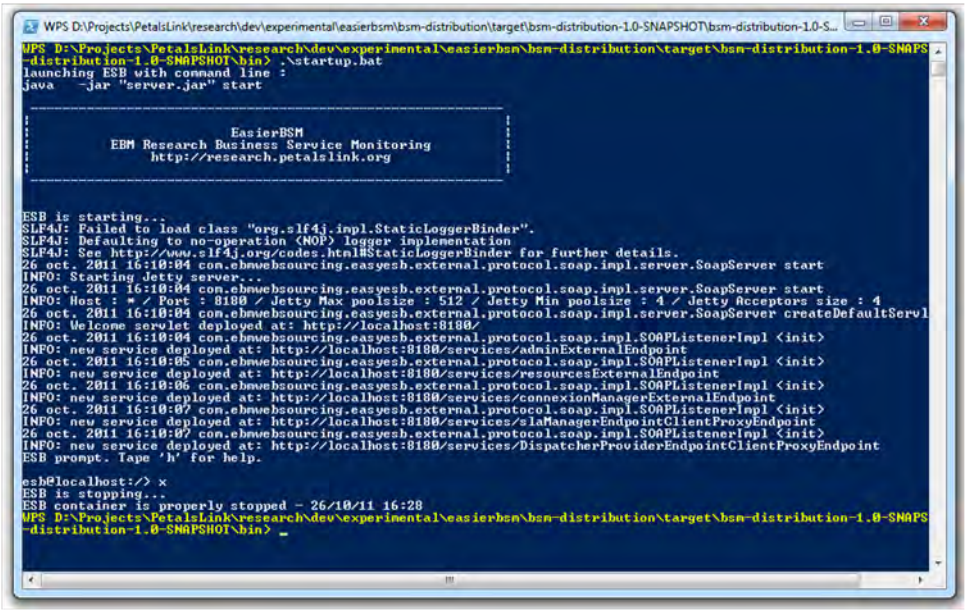

Figure 3.12: BSM Stopped Screenshot

4 CHOReOS IDRE Supporting tools

The CHOReOS Team relies on several tools for supporting their development within the project. These components are not considered as CHOReOS results. In this section, we present the installation and usage guides of two tools useful for both design and run-times. First, the Magic Draw tool is a No Magic tool dedicated to the modeling of processes, diagrams, workflows, etc. Within CHOReOS, we use this tool for designing the architectural specification of the CHOReOS components, middleware, governance framework, and IDRE. Moreover, in the scope of specific governance activitieswe extend its functionality with further improvements such as for the non functional annotations of choreographies for instance. Second, the EasiestDemo tool is a Linagora tool that provides useful functionalities for the runtime execution, testing, validation and packaging of SOA applications. It is helpful for supporting the development of EasyESB (See Section 3.2.4), EasierBSM (BSM Prototype, See Section 3.6.3) and EasierGov (Prototype of the SLA&Lifecycle Manager See Section 3.5.6).

4.1. Magic Draw

This section describes how to install CHOREOS IDRE supporting tool MagicDraw which is used for modeling choreographies. To fit supporting tool for CHOReOS IDRE solution you will need to follow these install guidelines:

Install Guidelines

```
   CHOReOS Install Files Template

Magic Draw Tool - INSTALLATION GUIDE - 08/10/2012

Written by Rokas Bartkevicious (No Magic)

== WHAT IS IT? ==
Magic Draw is a modeling platform. Within CHOReOS, we use specifically the Cameo Business Modeler
    Plugin for BPMN Choreographies diagram support,
Q4BPMN (Quality for BPMN) Profile for annotating the BPMN Choreography Diagram with quality
    requirements
and SysML Plugin - for defining requirements.

All MagicDraw and other plugins documentation you can find in MagicDraw tool (go to menu Help ->
    MagicDraw User Manual or Help -> Other Documentation) or in the internet http://www.nomagic.
    com/support/documentation.html

== REQUIREMENTS ==
System requirements for installing MagicDraw are described in the tool vendor No Magic page http:
    //www.nomagic.com/support/system-requirements.html#magicdraw-current


== CONFIGURING IT ==
Go to MagicDraw and download latest MagicDraw version:
https://www.magicdraw.com/download

To get the license for MagicDraw and required plugins you should request per tool vendor's e-mail
    sales@nomagic.com

After downloading and installing MagicDraw you need to download and install additional plugins.

Install Cameo Business Modeler (CBM) Plugin and SysML Plugin following the same steps for both as
    described following for SysML plugin http://www.nomagic.com/support/installation-and-use/
    plugins-and-profiles-install/sysml-plugin.html.
For Cameo Business Modeler (CBM) Plugin instructions are the same only the name is different.
```

```
== INSTALLING IT ==
To install and run MagicDraw follow these steps:
http://www.nomagic.com/support/installation-and-use.html#installing_and_running

==INSTALLING Q4BPMN Profile==
To install Q4BPMN Profile you should go to the Q4BPMN page and download resource:
http://labse.isti.cnr.it/tools/q4bpmn

How to install resource you can find here (check the topic "Installing profile if you already have
     the <profile name>.zip downloaded") http://www.nomagic.com/support/installation-and-use/
    plugins-and-profiles-install.html#profiles
```

4.2. Easiest Demo

This framework is based on several modules. A client module that provides a graphical user interface able to invoke Web Service like SoapUI. Moreover, it supports the realization of a complete SOA demonstration creating or importing EasyESB Nodes, EasierBSM Monitoring and EasierGOV Governance. A plugin (maven-wsoui-plugin) that allows users to generate a Web Service from a WSDL using wsdl2java module or to generate an SOA application using bpel2java module. This plugin is based on CXF Apache maven pluginsdk that allows user to create, compile and run an SOA application either using a graphical interface or using shell command. Finally, a gallery module that contains a set of use cases classified into three categories: business, academical or research.

Install Guidelines

```
   CHOReOS Install Files Template

Easiest Demo - INSTALLATION GUIDE - 08/10/2012

Written by Amira Ben Hamida (Linagora)

nicolas.salatge@linagora.com
julien.lesbegueries@linagora.com
amira.ben-hamida@linagora.com

== WHAT IS IT? ==
The Easiest Demo tool is delivered as part of the WP4 prototype. The prototype implementing the
SLA & lifecycle manager is called EasierGov. It ensures the management of the lifecycle of
    resources such services,
 SLA, choreographies, etc. Moreover, it provides an API enabling to synchronize with a service
     infrastructure (such as EasyESB- XSA)
 that allows managing services as WSDL and service level agreements as WS-Agreement.

== REQUIREMENTS ==
  1. Download java jdk 1.6
  2. Set JAVA_HOME in your classpath
  3. Copy this endorsed directory in jre library:
  $JAVA_HOME/jre/lib (for linux/windows) and $JAVA_HOME/lib (for mac)
  4. Download maven 2.2.1
  5. Set MAVEN_HOME in your classpath
  6. Set MAVEN_OPTS=-Xms512m -Xmx1024m -XX:PermSize=256m in your classpath

== CONFIGURING IT ==
 1.Download the latest release of EasiestDemo from this address:
http://research.petalslink.org/display/easiestdemo/Client+Binaries
 2.install it:
  On Windows OS, click on installer
  On Linux OS, java -jar EasiestDEMOClient-installer-OS-VERSION.jar

== COMPILING IT ==
Not required. The source code is available here:
http://research.petalslink.org/display/easiestdemo/Source+Code
Login: anonymous
Password : anonymous
```

```
== RUNNING IT ==
Go to the bin directory of project home (shortcut for users of the Windows OS):
 1.On Windows OS, start: ./startup.bat
 2.On Linux OS, start: ./startup.sh
 (remember to grant the write access on this file: chmod a+x startup.sh)

== VERIFYING IF IT WORKS ==
When started successfully the console presented in the figure
(EasiestDemo Started Screenshot) appears.
This tool is stopped by hitting the 'X' button of the console.
```

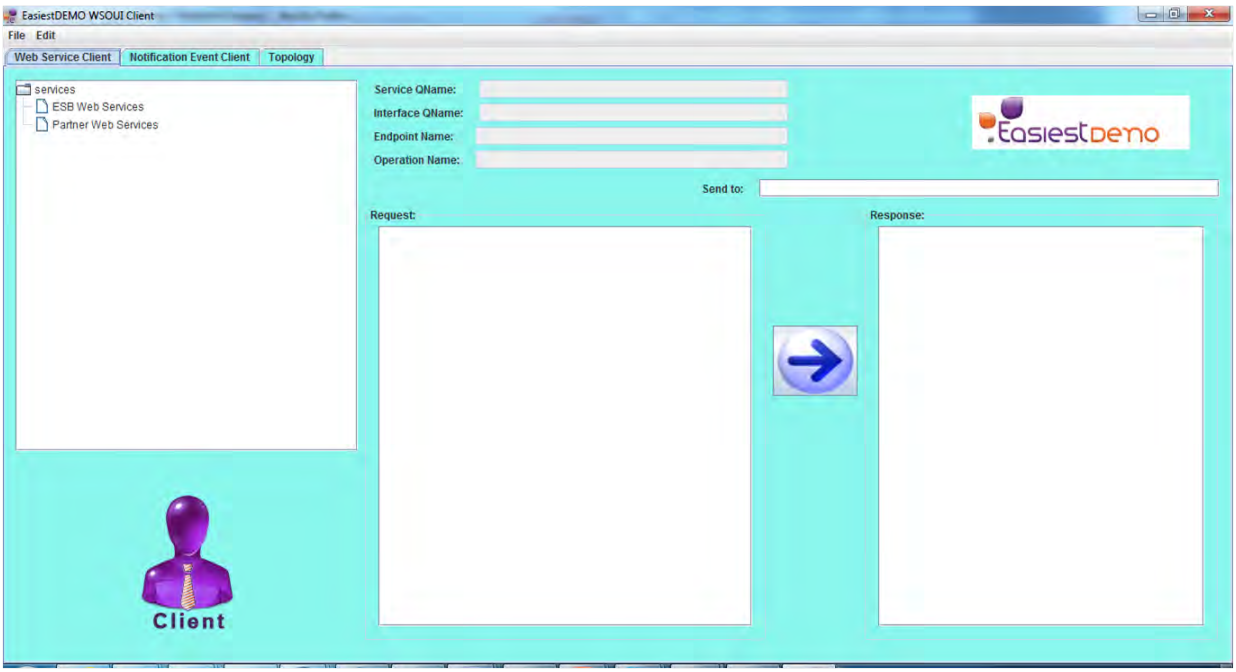

Figure 4.1: EasiestDemo Started Screenshot

The invocation of web service is realized in 7 steps:

1) Run the client application as in figure 4.1
2) Click on: File Add WSDL as shown in figure 4.2
3) Set Wsdl url as shown in figure 4.5
4) Once the service description is loaded, choose the operation to invoke by double-clicking as shown in figure 4.4.
5) Set the value of the operation parameters on the generated soap request (See Figure 4.5).
6) Call the web service by clicking on ”play” button (See Figure 4.6).
7) Wait provider response or error (See Figure 4.7).

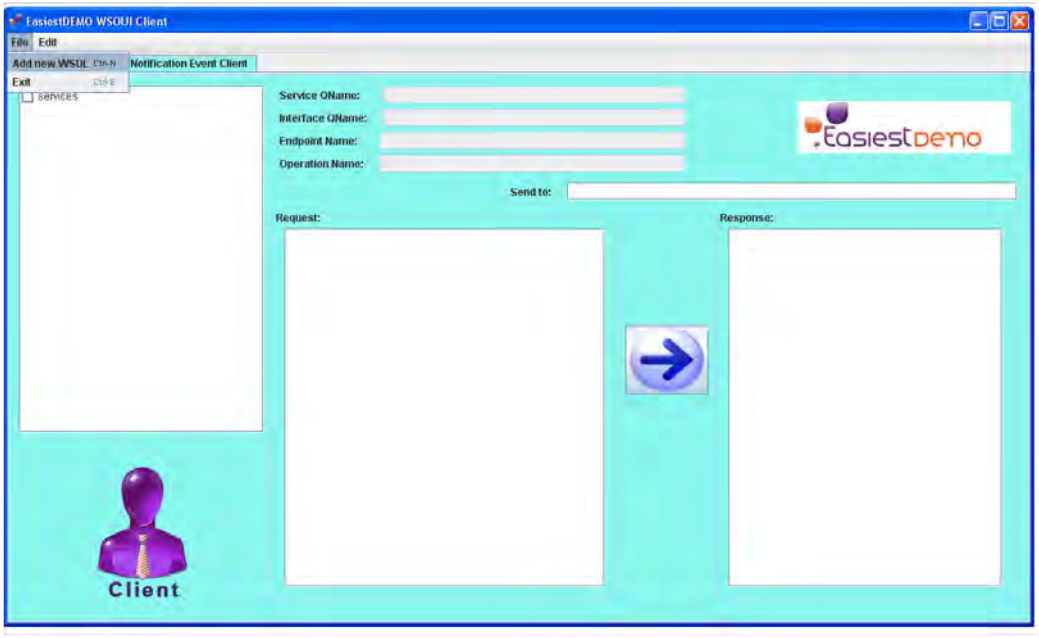

Figure 4.2: EasiestDemo Adding a Web Service

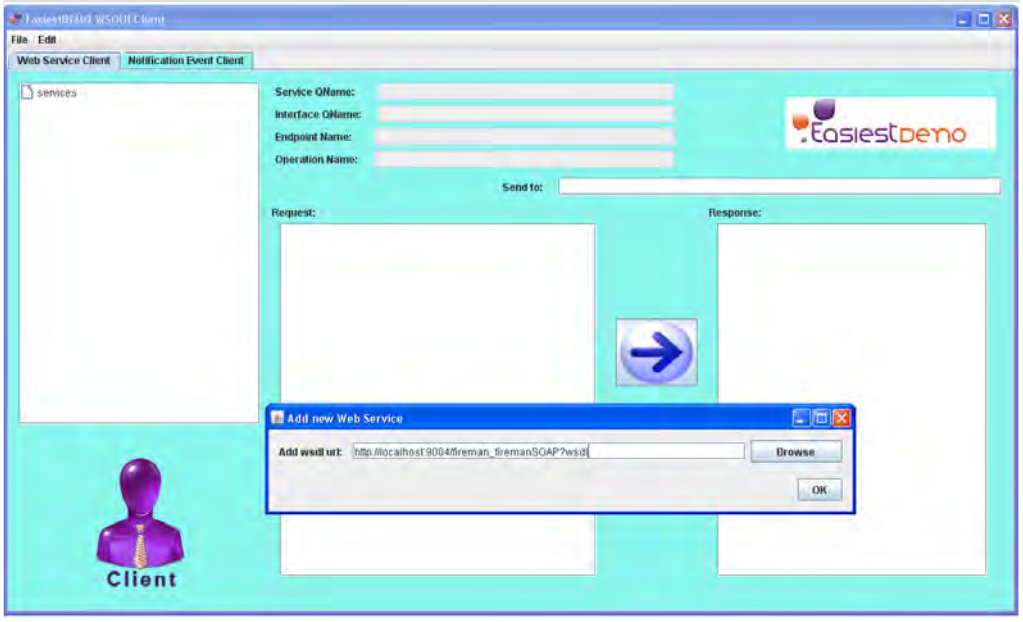

Figure 4.3: EasiestDemo Setting a WSDL

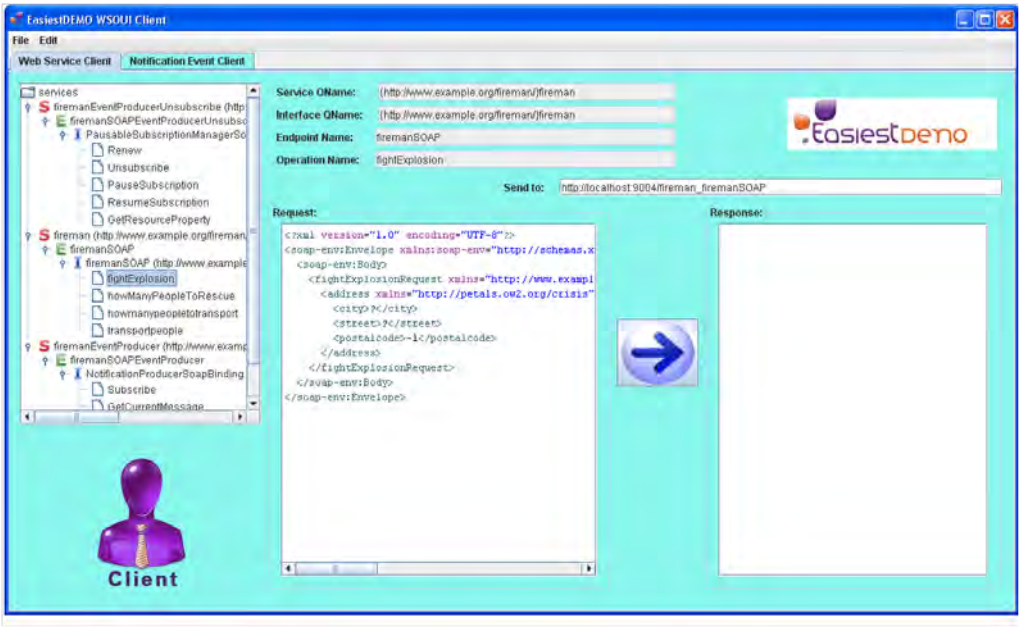

Figure 4.4: EasiestDemo Choosing an operation

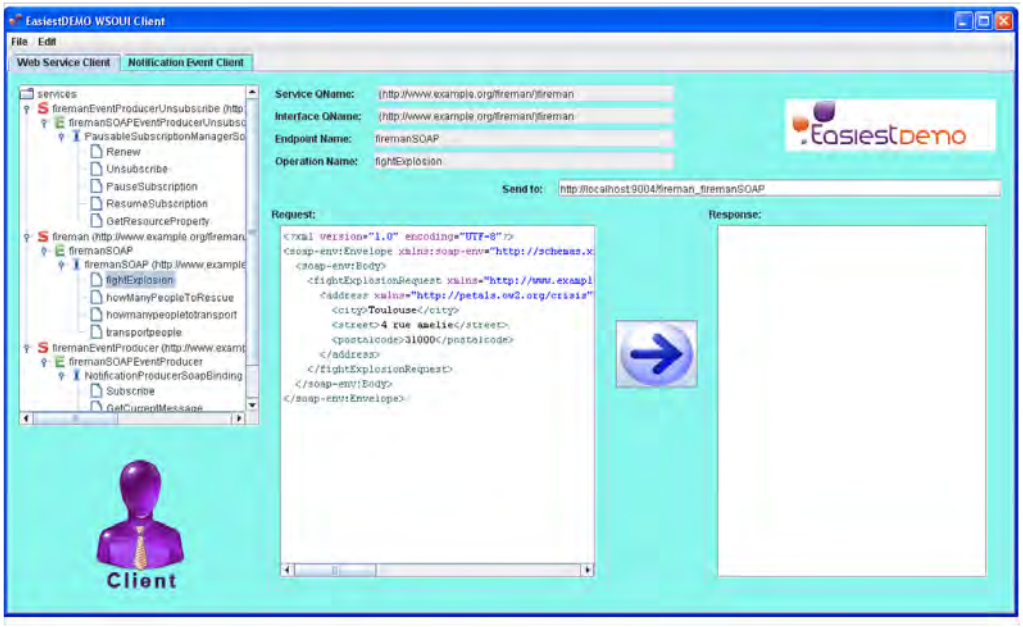

Figure 4.5: EasiestDemo Setting an operation

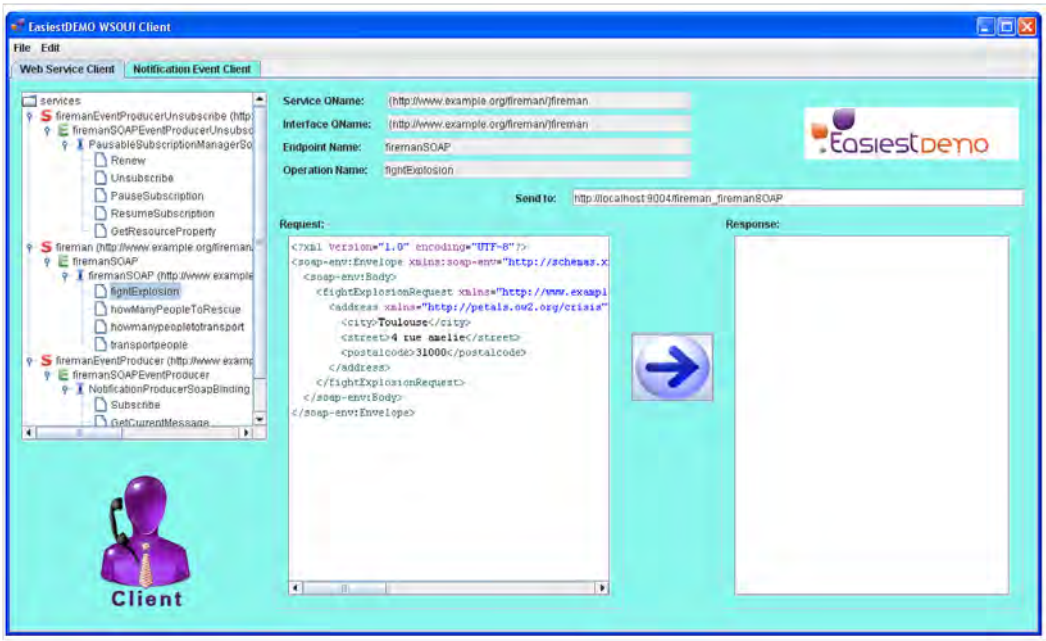

Figure 4.6: EasiestDemo Calling a Web Service

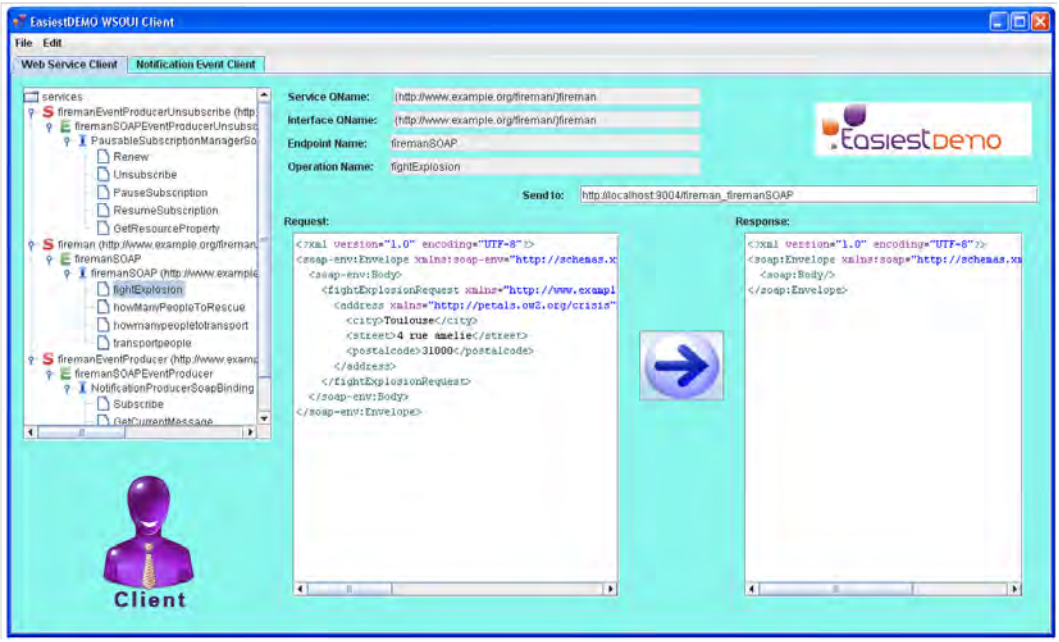

Figure 4.7: EasiestDemo Calling a Web Service

5 Conclusion

In this deliverable, we present the first version of the CHOReOS IDRE as well as the available components. The IDRE is accompanied with the technical usage manuals describing the steps to be followed for using the components. These user manuals are dedicated to the CHOReOS developers and to users with technical knowledge. More elaborated end user manuals need to be delivered at Year 3 with the release of the open source packages and libraries as planned in the DOW. In the following section 5.1, we recall the different components as well as their locations. Next, in section 5.2 we describe the coming steps during the period M24 to M30 of the project.

5.1. Components Availability

Subsystem	Component	Repository Location
XSC	Component-cd	choreos/trunk/executable_service_composition/dsb_cd_implementation
	C&E	choreos/trunk/executable-service-composition/c-and-e/
XSA	XSB	choreos/trunk/extensible-service-access/xsb-over-dsb
	Sensor Access Middleware	choreos/trunk/extensible-service-access/lsb/sensorAccess
	Phone Proxy Services	choreos/trunk/extensible-service-access/
	EasyESB	http://research.petalslink.org/display/easyesb/Downloads
XSD	AOSMB	choreos/trunk/extensible_service_discovery/AoSBM
	Discovery Plugin Framework	choreos/trunk/extensible_service_discovery/plugin_manager
	Things Discovery	choreos/trunk/extensible_service_discovery/
Cloud and Grid	Node Pool Manager	choreos/trunk/cloud/NodePoolManager
	Storage Factory	choreos/trunk/cloud/ServiceFactory
	Service Deployer	choreos/trunk/cloud/ServiceDeployer
	Grid	choreos/trunk/grid/integrade
Governance and V&V Framework	Rehearsal	governance/tdd/rehearsal
	Service Pot	governance/servicepot
	Partes	governance/v_and_v/partes
	CRank	governance/v_and_v/crank
	SLA&LifecycleManagement	governance/sla_and_lifecycle_manager
Monitoring Framework	Glimpse	governance/component-glimpse
	Business Service Monitoring	governance/new-bsm-distribution-choreos
	Platform Monitoring	cloud/Monitoring

5.2. Next Steps

This deliverable as well as the implemented test bed presented in D5.4 and the 2nd version of the integration plan D5.7.2, pave the way to the deployment and use of the CHOReOS IDRE as a OSS platform, that will be achieved at M30 of the project. Indeed, we consider that a first step towards this is to clearly identify the components taking part in the IDRE. Second, relying on the collaborative development tools set up within WP5 activities and to the code provided by the CHOReOS team, we aim at achieving the successful integration and assessment of each component based on the unitary tests and the use case. Third, the implemented use cases will benefit from the functionality of the IDRE as a unique environment.

Bibliography

[1] Rokas Bartkevicius, Amira Ben Hamida, Guglielmo De Angelis, and Darius Silingas, editors. *CHOReOS whitepapers*. Number Del. D9.4. The CHOReOS Consortium, April 2012. www.choreos.eu.

[2] Cesare Bartolini, Antonia Bertolino, Andrea Ciancone, Guglielmo De Angelis, and Raffaela Mirandola. Non-functional analysis of service choreographies. In *Proceedings of the 4th International Workshop on Principles of Engineering Service-Oriented Systems (PESOS), 34th International Conference on Software Engineering (ICSE)*, Zurich, Switzerland, June 2012. IEEE-CS.

[3] Cesare Bartolini, Antonia Bertolino, Andrea Ciancone, Guglielmo De Angelis, and Raffaela Mirandola. Quality requirements for service choreographies. In *Proceedings of the 8th International Conference on Web Information Systems and Technologies (WEBIST)*, Porto, Portugal, April 2012.

[4] A. Bertolino, G. De Angelis, and A. Polini. Validation and verification policies for governance of service choreographies. In *Proc. of the 8th International Conference on Web Information Systems and Technologies (WEBIST 2012)*, Porto, Portugal, Apr. 2012. SciTePress.

[5] CHOReOS Project Team. D5.2: Choreos idre specification. Technical report, April 2011. www.choreos.eu.

[6] CHOReOS Project Team. D2.2: Definition of the dynamic development process for adaptable qos-aware uls choreographies. Technical report, April 2012. www.choreos.eu.

[7] CHOReOS Project Team. D3.2.1: Choreos middleware implementation first version. Technical report, July 2012. www.choreos.eu.

[8] CHOReOS Project Team. D3.2.2: Choreos middleware implementation. Technical report, October 2012. www.choreos.eu.

[9] CHOReOS Project Team. D4.2.2: V&v tools and infrastructure strategies, architecture and implementation. Technical report, April 2012. www.choreos.eu.

[10] Antinisca Di Marco, Claudio Pompilio, Antonia Bertolino, Antonello Calabrò, Francesca Lonetti, and Antonino Sabetta. Yet another meta-model to specify non-functional properties. In *Proceedings of the International Workshop on Quality Assurance for Service-Based Applications*, QASBA '11, pages 9–16, New York, NY, USA, 2011. ACM.